Universidade do Minho
School of Engineering

José Diogo Cruz de Moura

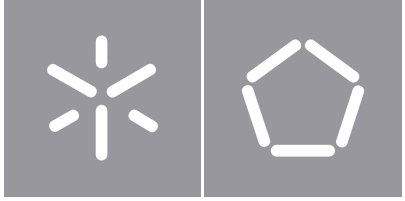
Development and integration of topology–based methods for gap-filling of metabolic networks

october 2023

Development and integration of topology-based methods for gap-filling of metabolic networks

José Diogo Cruz de Moura

UMinho | 2023



University of Minho
School of Engineering

José Diogo Cruz de Moura

Development and integration of topology--based methods for gap-filling of metabolic networks

Masters Dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Óscar Dias

october 2023

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

Não posso deixar de começar com um enorme agradecimento a Emanuel e João Capela. Foram os dois a inspiração para o tema deste trabalho. Ajudaram-me muito e sempre muito recetivos e proativos. Agradeço, Emanuel, pela tua dedicação contínua e pelo que me ensinas constantemente, este trabalho tem uma costela muito grande tua. Agradeço ao professor Óscar e aos outros docentes de Bioinformática por terem aparecido e contribuído no meu percurso científico, onde aprendi muito.

Agradeço ao Capela, pela tua visão objetiva e organizada. De caderno na mão a simplificar o complexo. Mas, acima de tudo, agradeço as conversas e a amizade que temos fora do âmbito académico e espero que a nossa ligação se perpetue e que tenhamos muitas mais conversas sobre temas fraturantes.

Agradeço aos meus dois grandes amigos, Camila e Tomás, por tudo o que me dão. Camila, por todas as refeições e convívios em tua casa e pela tua simplicidade em nutrir a nossa amizade e por seres puramente bondosa. Tomás, por estares sempre disponível para ouvir os meus novos dilemas e problemas, e por termos a melhor amizade que podia imaginar. Vocês os dois sabem o quão grato estou pela vossa existência na minha vida.

Agradeço especialmente à Constança, por me suportar constantemente, pelo carinho e amor que existe, pela companhia, pelo resistir aos problemas, pela ligação de mutualidade, pela presença inabalável.

Agradeço aos meus pais e irmã, por me ajudarem sempre que preciso para qualquer situação e por me aturarem quando estou mal humorado.

Às restantes pessoas, família e amigos, que fazem parte da minha vida, e me apoiam de diferentes formas, e que me vêem terminar mais uma fase da minha vida, deixo aqui a minha gratidão.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, december 2023

José Diogo Cruz de Moura

Abstract

The usage of Genome-scale Metabolic models (GEMs) spans various applications across diverse fields. These models depict a metabolic network that represents the complete metabolism of a specific organism. The automated reconstruction of these models can be facilitated using tools like *merlin*. However, the draft models generated often contain gaps, primarily due to knowledge limitations in the databases that enable the reconstruction of these models (e.g., KEGG, ModelSEED), and due to wrong and faulty genome annotations. Certain tools, such as BioISO, aim at identifying metabolites that cannot be produced by a metabolic network. Other tools like Meneco try to discern a set of reactions that can be integrated into the model, aiming to rectify the existing gaps in the metabolic network. The execution time for these tools increases proportionally with the complexity of the metabolic network under scrutiny, potentially leading to prolonged execution times to address the possible gaps present in the model. Thus, there arises a need to develop a workflow that is both efficient and offers reliable solutions. In this study, a workflow integrating BioISO and Meneco (BioMeneco) was developed, coupled with the development of pertinent methods, with the aim of automating the process as much as possible, reducing the search space, and optimising the gap-filling process. The outcomes of the developed workflow were promising. It not only offered reduced execution times but also provided the capability for better refinement for various models when compared to a typical Meneco workflow. Regardless, while the developed workflow demonstrated efficiency, it also highlighted the challenges of relying on a single database and the complexities of metabolic networks, paving for further improvements and research in this domain.

Keywords Gap-filling, Genome-scale Metabolic Model, Metabolic Model Reconstruction, Optimisation, Python Implementation, Systems Biology, Workflow Development

Resumo

O uso de Genome-scale Metabolic models (GEMs) tem diferentes aplicações para diversas áreas. Estes modelos representam uma rede metabólica que espelha o metabolismo integral de um dado organismo. A reconstrução automática destes modelos pode ser efetuada com o auxílio de ferramentas como o *Merlin*. Contudo, os *draft models* criados apresentam *gaps* devido à falta de conhecimento nas bases de dados que possibilitam a reconstrução destes modelos (por exemplo, KEGG, ModelSEED). Existem algumas ferramentas (BioISO) que visam identificar metabolitos que não são produzidos numa dada rede metabólica. Após à identificação destes metabolitos, outras ferramentas, como o Meneco, utilizam esta informação para descobrir um conjunto de reações que possam ser incorporadas no modelo, visando corrigir os *gaps* existentes na rede metabólica. O tempo necessário para executar estas ferramentas aumenta proporcionalmente com a complexidade da rede metabólica em análise, o que pode resultar em tempos de execução muito altos para mitigar os possíveis erros presentes no modelo. Deste modo, floresce a necessidade de desenvolver um *workflow* que seja eficiente e com soluções fiáveis. Neste estudo, foi desenvolvido um *workflow* que integra o BioISO e o Meneco (BioMeneco), aliado ao desenvolvimento de métodos relevantes, com o objetivo de ser um processo o mais automático possível, reduzir o espaço de procura e otimizar o processo de *gap-filling*. Os resultados do *workflow* desenvolvido foram promissores, oferecendo tempos de execução decentemente reduzidos, e maior capacidade de refinamento, para diferentes modelos, quando comparados com um *workflow* típico de Meneco. No entanto, apesar do *workflow* desenvolvido ter demonstrado eficiência, também evidenciou os desafios de depender de uma única base de dados e as complexidades das redes metabólicas, abrindo caminho para futuras melhorias e investigação nesta área.

Palavras-chave Biologia de Sistemas, Desenvolvimento de *Workflow*, *Gap-filling*, Genome-scale Metabolic Model, Implementação em Python, Otimização, Reconstrução de Modelos Metabólicos, Redes metabólicas

Contents

List of Figures

List of Tables

Abbreviations List

merlin Metabolic Models Reconstruction using Genome-Scale Information. 1, 2, 3, 8, 14, 15, 16, 26, 29, 46, 47, 48, 51, 56, 64

ACP Acyl-carrier protein. 49

ASP Answer Set Programming. 18, 19, 21, 25

ATP Adenosine Triphosphate. 11, 31

BioISO Biological networks constraint-based In Silico Optimization. 2, 25, 26, 27, 29, 34, 35, 45, 46, 52

BLAST Basic Local Alignment Search Tool. 8

CPLEX IBM ILOG CPLEX Optimization Studio. 34, 45

DNA Deoxyribonucleic Acid. 8, 10, 28

EC Enzyme Comission. 8, 9, 13

FBA Flux Balance Analysis. 8, 12, 24, 25, 26, 43

FCA Flux Coupling Analysis. 18

FVA Flux Variability Analysis. 12

GEM Genome-scale Metabolic Model. 1, 2, 6, 8, 12, 13, 15, 16, 26, 29, 47, 48

GLPK GNU Linear Programming Kit. 34

GPR Gene-Protein-Reaction. 9, 13, 14, 15, 16

ILS Integer least squares. 19

JSON JavaScript Object Notation. 30, 31, 40, 42, 44, 45, 59

KEGG Kyoto Encyclopedia of Genes and Genomes. 8, 9, 13, 20, 28, 29, 43, 45, 48, 49, 55, 57, 58, 61, 63

LP Linear Programming. 19

MILP Mixed Integer Linear Programming. 19

NCBI National Center for Biotechnology Information. 8

NGS Next-generation sequencing. 4

OS Operating System. 44

pFBA Parsimonious enzyme usage FBA. 12

RNA Ribonucleic Acid. 8, 10

SBML Systems Biology Markup Language. 13, 15, 16, 28, 30, 36, 39, 40

TC Transporter Classification. 8, 9

TCDB The Transporter Classification Database. 9

tRNA Transfer RNA. 10, 11, 33

XML Extensible Markup Language. 43, 44, 45

Chapter 1

Introduction

1.1 Context and motivation

Genome-scale Metabolic Models (GEMs) are tools currently used to understand and analyse the physiological and metabolic characteristics of a given organism under different environmental and genetic conditions. These models reveal great potential in predicting biological capabilities, metabolic engineering, and systems medicine (4; 5; 6; 7). The reconstruction of a GEM is typically generated using software platforms, such as Metabolic Models Reconstruction using Genome-Scale Information (*merlin*) (2; 8). This tool includes several key features, including functional genomic annotations, identification and annotation of transport protein genes, generation of transport reactions, semi-automatic enzymatic re-annotation, and determining GPR associations (2). One of the main reasons for the divergence in phenotype predictions using GEMs and experimental data is the existence of gaps in metabolic networks (4). It is important that these models are refined and evaluated through various stages to ensure their quality. This includes the processes of gap-finding and gap-filling. Gap-finding helps identify potential deficiencies such as missing reactions, unknown pathways, unannotated or misannotated genes, as well as promiscuous enzymes and underground metabolic pathways. Gap-filling addresses these issues by incorporating the missing information, thereby resolving these gaps (4). Recent gap-filling algorithms, adopt novel techniques based on machine learning, network topology analysis and likelihood modelling. Some examples are Meneco (1) and BoostGAPFILL (9). Although computational gap-fillers demonstrate being able to identify and add a significant number of correct reactions, manual curation is still required to obtain high-accuracy models (10). However, while tools like Meneco (1) have been instrumental in advancing the field of metabolic network reconstruction, these still have their own limitations. One of the primary challenges with Meneco (1) is its computational time, especially when dealing with large-scale metabolic networks. As the complexity of the model increases, Meneco can become time-consuming, making it less feasible for real-time or iterative model refinements. Furthermore, Meneco's (1) topological approach, while robust, might

not always capture the intricate dynamics of certain metabolic pathways, leading to potential oversights in gap-filling. To address the aforementioned challenges, Biological networks constraint-based In Silico Optimization (BioISO) (11) presents a promising approach. It efficiently identifies blocked reactions and dead-end metabolites within metabolic networks (11). In this context, the combination of (11) and Meneco (1) with other relevant methods can significantly enhance the gap-filling process, offering a comprehensive solution to the challenges faced in metabolic model refinement, ensuring a more accurate representation of biological systems. Thus, it is of great interest to develop and implement novel gap-filling methods, in the reconstruction of GEMs, in order to ensure the existence of high-quality GEMs.

1.2 Goals

More general objectives related to scientific investigation would be the review of state-of-the-art subjects related to metabolic models, the automatic generation of draft models and the technical and theoretical intersection of these. As well as, analysing existing tools for topological gap-filling.

To obtain high-quality GEMs, the gap-filling process must be enhanced with novel methods. BioMeneco emerges as a promising starting point for developing a comprehensive workflow. This dissertation concentrates on the creation of such a workflow, incorporating novel methods and approaches. Specifically, this workflow is designed to encapsulate BioMeneco with the developed methods, aiming to produce a more accurate and efficient process than the typical standalone workflows. Upon execution, a comprehensive report detailing the results would be generated, providing insights into the efficacy and outcomes of the integrated workflow.

The ultimate objective is to operationalise this developed workflow within *merlin* (2), either as a standalone operation or as a plug-in.

1.3 Documentation outline

The document is organised as follows:

1. **Chapter 2**: State-of-the-art
    (a) a simple description of systems biology
    (b) detailing the reconstruction steps of a GEM model and its automatic process
    (c) exploration and description of the gap-filling concept

(d) overview of existing gap-filling tools, especially topology-based ones

(e) overview in more detail of Meneco and BioISO

2. **Chapter 3**: Materials and Methods

(a) explanation of the generation of a draft model with *merlin*

(b) description of how seeds and targets are identified within the developed workflow

(c) detailed method for the creation of a Costum Universal Model

(d) utilisation of Docker for software containerisation and its significance in the workflow

(e) overview of performance tests conducted for the developed workflow, specifically using BioMeneco

(f) brief description of the generated files and their significance in relation to the developed workflow

3. **Chapter 4**: Results and Discussion

(a) overview and statistical summary of the utilised *Streptococcus pneumoniae* R6 gap-induced model and the draft models generated with *merlin*

(b) in-depth analysis of the performance test results for different draft models and gap-induced models using the developed workflow

4. **Chapter 5**: Conclusions and Future work

(a) insights into the potential impact and contribution of the developed workflow, along with future research directions

Chapter 2

State-of-the-art

2.1 Systems Biology

In the last two decades, there has been a significant improvement in sequencing throughput and a decrease in its costs (12). Next-generation sequencing (NGS) has only been commercially affordable for several years, it has already contributed immensely to a great impact in different scientific and technological fields (12), including systems biology.

Through the success of the bio-sciences studying individual components (molecules, enzymes, genes, etc.), driven by bioinformatics, omics, and genome sequencing, there is a research focus that has moved from individual components to networks (13). Even if it was possible to study each component individually, it would still lack an understanding of how it functions in the context of a complex living organism. Such studies would not allow elucidating about supramolecular functional properties such as the cell cycle, metabolic steady states and cell (dys-)function, nor understanding multifactor diseases, white or green biotechnology (14). Moreover, molecular bio-sciences offer a lot of data that require system approaches to understand the functioning of the cell. To be able to understand these subjects a systems approach is required (14). Thus, the field of systems biology has emerged, which may be viewed as a way of thinking on how to look into a biological organism (15). There is a need to bring together several different areas of knowledge to adopt system approaches in bio-sciences. This implies, for instance, that a molecular bio-scientist is familiar with current mathematical and computational methods, and a mathematician is aware of acceptable biological mechanisms. Some reviews have been made that address the challenges inherent to the emergence of systems biology (14). According to Bruggeman and Westerhoff (2007) (14), there are two procedures for working in this field, which can help to simplify the way a subject is approached and make it more reliable: top-down and bottom-up systems biology.

The main purpose of top-down systems biology is to discover new biological knowledge using an iterative workflow. There is the attempt to observe the system under study from a whole, bird's eye view

- by measuring genome-wide data - having the ambition to discover or describe biological mechanisms close to the bottom - representing segments and interactions. It starts with experimental data, followed by data analysis and data integration to determine correlations, ending with hypothesis formulation. These hypotheses can predict new correlations, which can be tested by experiments or further analysis. Because there is an omics approach, the main advantages here are that there is a complete, genome-wide approach (14). It provides a more complete understanding of the interactions and connections within a biological system. Thus, new discoveries and insights can be revealed since this approach offers a multiple-dimension perspective.

On the other hand, bottom-up systems biology is a different approach to understanding, in terms of molecular interactions, how biological organisms work (14). The main premise points to studying each component of a whole system, and then integrating this knowledge to predict the behaviour of the target system. Thus, the major goal would be to create a comprehensive model that is able to explain the functioning of the entire organism. This approach relies on experimental studies related to kinetic and physico-chemical conclusions of the components, data regarding cellular responses to perturbations, the construction of models for target organisms and the development of tools to analyse and represent these models (14).

2.2 Metabolic Modelling

The metabolic modelling field is a crucial component in the realm of systems biology. Metabolic modelling is integral to the systematic analysis of biological systems, offering profound insights into the intricate network of metabolic pathways. It facilitates the prediction of metabolic phenotypes under various genetic and environmental conditions (16). This is achieved by using mathematical models that represent the genes, proteins, reactions, and metabolites of an organism, as well as the interactions between them. However, such models are developed based on available experimental data and biological databases which may not always be complete. Thus, it is common to find missing reactions in these models. Gap-filling is a computational procedure that proposes the addition of reactions to genome-scale metabolic models, ensuring these models are complete and interconnected. Such completeness is essential as models derived from annotated genomes often lack fully connected metabolic networks due to unidentified enzymes (10). Furthermore, understanding interactions, such as those between ageing and genetic variations that result in disease phenotype, is essential. The use of model systems, like the mouse, has proven invaluable in studying the relationship between ageing and metabolism, as well as the need for modelling these processes

(17). The gap-filling process, especially in the context of non-model organisms, has been instrumental in elucidating host-microbiota cooperation and understanding the metabolism of organisms (18).

2.2.1 Reconstruction of GEMs

Context

Metabolic models are partial or full mathematical representations of the metabolism of an organism (5; 19; 20). Since computational methods are used to perform these representations, they are known as *in silico* representations. These models can integrate different data, including genome sequences, omics data, and biochemical information found in biological databases (21). Furthermore, these models can also be used to simulate metabolic fluxes for various systems-level metabolic studies (21).

The first GEM was reconstructed in 1999 for *Haemophilus influenzae* (22). Since then, with new methods, the rise of systems biology approaches and high-throughput technologies, many other reconstructions have been made for different organisms (21).The cellular metabolism of an organism may be represented by a set of metabolites, reactions and constraints in these GEM, which can be viewed as a fully functional database (6; 23). These models describe a whole set of stoichiometry-based metabolic reactions of a target organism based on information retrieved from the genome sequence and experimentally data (21). Reconstructing a high-quality GEM is a complex and time-consuming process that can take months to complete, depending on the complexity of the target organism, the level of model curation desired and the tools used (24). The reconstruction process is well described in the literature (24; 25; 26). This procedure has been simplified by many authors, and is typically divided into four main phases: genome annotation, assembly of a metabolic network, conversion of the network into a stoichiometric model, and model validation (8; 26; 24).

There is an alternative top-down approach to reconstructing Genome-Scale GEMs. This method focuses on understanding overall cellular behaviour before delving into specific components, utilising advanced omics technologies. By analysing these data, researchers reverse-engineer the metabolic network based on observed cellular phenotypes (27). The top-down approach offers an alternative for GEM reconstruction, allowing models to be built based on the observed behaviour of the system rather than relying solely on genomic information. The choice between bottom-up and top-down approaches depends on the available data, the characteristics of the target organism, and the research goals.

Metabolic models as a computational representation

Metabolic models serve as computational representations that encapsulate the biochemical transformations occurring in a biological system. These representations are fundamentally grounded in the principles of stoichiometry and are structured as directed bipartite graphs, facilitating the systematic analysis of biological systems by offering insights into the complex network of metabolic pathways (28). A prominent tool in this field is COBRApy (29), a Python library that facilitates the analysis of metabolic networks through structured representations of metabolic models, aiding in the prediction of metabolic phenotypes under various genetic and environmental conditions (30; 31). This tool is highly utilised among different users, and its integration with many other tools is vastly compatible.

Below, the essential aspects of how COBRApy (29) structures these representations:

Definitions:

G : Graph representing the metabolic model, $G(V, E)$.

V : Set of vertices representing metabolites in the graph.

E : Set of edges representing reactions in the graph.

G' : Set of genes, metadata linked with reactions.

C : Set of compartments, representing distinct subgraphs within G.

Attributes:

For each $r \in E$, there may be additional attributes, such as $kcat$ (catalytic rate constant).

For each $m \in V$, there may be attributes, e.g., charge, formula, or compartment assignment.

For each $g \in G'$, there may be attributes related to gene expression or regulatory information.

Data Structures:

- **Reactions**: Defined as $r = (sub, prod, lb, ub, genes)$, where:

  sub : Set of substrate metabolites

  $prod$: Set of product metabolites

  lb, ub : Flux bounds as real numbers

  $genes$: Associated genes

- **Metabolites**: Defined as $m = (ID, formula, compartment)$.

- **Genes**: Linked with reactions, defined as $g = (ID, reactions)$.
- **Objective Function**: Mathematically represented as:

$$obj = \sum_{r \in R} c_r \times flux(r)$$

where c_r is the coefficient of reaction r and $flux(r)$ is the flux value.

Additionally, COBRApy (29) leverages Flux Balance Analysis (FBA) (32) to predict metabolic phenotypes by computing genome-scale flux distributions that optimize a cellular fitness objective, typically the cellular growth rate. This objective is modelled as a linear combination of synthesis rates of various biomass components, such as amino acids and lipids, based on the constraints imposed on each metabolic flux, allowing for efficient problem-solving through linear programming algorithms (33; 34).

Genome Annotation

To establish a foundational understanding of computational representations, the subsequent sections will delve into the intricacies of reconstructing a GEM. Genome annotation involves two key steps: functional annotation and structural annotation. Structural annotation identifies regions of Deoxyribonucleic Acid (DNA) that encode protein products, various types of Ribonucleic Acids (RNAs), and other relevant features (35). On the other hand, functional annotation provides insights into biological functions intrinsic to the genome (35). Enzyme Comission (EC) (36) and Transporter Classification (TC) (37) numbers are commonly employed to assign functions to enzymes and transport proteins in a standardized and objective manner.

Obtaining genome annotations represents the initial step in the reconstruction of GEMs. Databases such as Kyoto Encyclopedia of Genes and Genomes (KEGG) (38) and National Center for Biotechnology Information (NCBI) (39) store this type of information. While quality annotations are crucial for an accurate metabolic network representation (24), the high cost and time consumption associated with experimental verification make bioinformatics methods the preferred approach. Various bioinformatics tools, including gene-finding algorithms like GLIMMER (40) and AUGUSTUS (41), facilitate genome structural annotation.

In the absence of reliable functional annotation, sequence similarity alignment techniques such as Basic Local Alignment Search Tool (BLAST) (42), HMMER (43), or DIAMOND (44) can be employed.

It's important to note that the assignment of EC and TC numbers occurs throughout the annotation process. While tools like *merlin* (2) enable semi-automatic or fully-automatic annotation, manual verification becomes necessary during the reconstruction of a high-quality GEM. The accuracy of the model may

be significantly impacted by incorrect annotations, potentially leading to the exclusion of certain reactions and the emergence of gaps in the model (45; 46; 24).

Network assembly

The second step concerns the assembly of the metabolic network, which involves several minor steps. The EC and TC numbers identified in the genome annotation may be used to access enzymatic and transport reaction data from databases, such as KEGG (47) and BRENDA (48). Regarding the TC number, it can also be used to retrieve transport reaction data from the The Transporter Classification Database (TCDB) (49). The assemble of transport reactions directly from this database is not straightforward. Hence, it is recommended to use a tool design for this purpose, like TranSyT (50) Information such as protein names, EC and TC numbers, reaction identifiers, stoichiometric equations, reactants, and products, is retrieved from a database and assembled together to generate a draft network. Spontaneous and non-enzymatic reactions are added to complete the draft metabolic network using literature or databases. The next task is to assign the reactions to the corresponding sub-cellular organelle. In case there are similar reactions occurring in different compartments, it is needed to specify a fixed identifier corresponding to the reaction and metabolite, for the respective compartment (24). This is particularly pertinent for complex organisms, where higher complexity often entails an increased number of compartments. Knowledge of protein location can be found in literature and databases, namely, UniProt (51) and BRENDA (48). Some tools, including TargetP, (52), LocTree3 (53) and WoLF PSORT (54), use the sequence to predict the sub-cellular compartment for enzymes.

Then, metabolic genes must be associated with proteins and reactions through Gene-Protein-Reaction (GPR) associations, which are represented using boolean rules (AND/OR). Here the involvement of each gene and its associated product is linked to biochemical reactions (55). The simplest case is when a gene encodes a protein that catalyses a reaction. However, more complex GPRs can be found due to the existence of isoenzymes, promiscuous enzymes, and protein complexes (24). The existence of these is taken into account in the metabolic network, to assure the maximum depiction of reality (24; 55).

A number of procedures must be followed as a manual refinement process and curation, including the specificity of the substrate and cofactors, mass balance, the reversibility of the reactions, stoichiometry and gap-filling (24). Missing reactions, unidentified pathways, incorrectly and incompletely annotated genes, promiscuous enzymes and underground metabolic pathways can all be found by gap-filling investigations (4; 45).

Conversion into a stoichiometric model

A reaction that represents the biomass formation must be created at this stage - $equation\ 1$ - this, will characterise all biomass components that are known to be present in the organism. This reaction characterises the contribution of the components to the cellular biomass. As a consequence, and because macromolecules are associated with biomass components, the biosynthesis processes of these molecules have to be included in the network (24).

$$equation\ 1:$$

$$\sum_{k=1}^{p} c_k X_k \rightarrow \text{Biomass},$$

where c_k are the coefficients of the metabolites and X_k are the metabolites.

These components typically encompass DNA, RNA, lipids, proteins, carbohydrates, cell wall components (if applicable) and cofactors. Other components can be added depending on the target organism.

The flux of this reaction indicates the organism's growth rate (45). There is the necessity to experimentally calculate the biomass of the organism under study, which is calculated for cells in the logarithmic phase (24; 56). In cases the experimental data is not available, bioinformatics tools and literature can be used to estimate the biomass instead. When this is not available, biomass composition from experiments on closely related organisms may be used instead (57).

Since an organism usually has a high number of biomass components, the biomass reaction is often split into several reactions representing the production and assembly of each macromolecule. For instance, a reaction representing RNA formation is formulated as follows:

$$equation\ 2:$$

$$aATP + bUTP + cGTP + dCTP \rightarrow RNA + sDiphosphate$$

where a, b, c, and d are the molar composition of each RNA component in $mmol/gRNA$, s is the sum of a, b, c and d. The RNA metabolite is then included in the biomass main reaction.

A similar procedure is followed for the remaining macromolecules. The protein biosynthesis is represented using Transfer RNAs (tRNAs) charged with amino acids:

$$equation\ 3:$$

$$\sum_{k=1}^{20} c_k X_k \rightarrow \text{Protein} + \sum_{k=1}^{20} c_k Y_k + sH_2O,$$

where c_k are the coefficients of the metabolites, s is the sum of the molar composition of all amino acids, X_k are the aminoacyl-tRNA (tRNA charged with each amino acid), and Y_k are the tRNAs not charged with each amino acid.

The aminoacyl-tRNA biosynthesis occurs using Adenosine Triphosphate (ATP), a free tRNA, and the specific amino acid:

$$equation\ 4:$$

$$ATP + AA + trna(AA) \rightarrow AA - tRNA + AMP + Diphosphate$$

where AA is an amino acid, $trna(AA)$ is a free tRNA, and $AA - tRNA$ is an aminoacyl-tRNA.

Growth-associated energy is often included in the biomass formulation to account for the energy that the organism uses to produce biomass macromolecules. The growth-associated energy characterizes the amount of ATP needed per gram of biomass produced (45). On the other hand, non-growth-associated maintenance energy (e.g., pH maintenance) has to be included in an ATP hydrolysis reaction. These energy requirements can be found in literature or retrieved from experimental data. In the case of experiments, attempts to plot ATP production against biomass growth rate, where y-intercept refers to non-growth associated maintenance and slope to growth associated maintenance (58). If this is not possible, this parameter can be estimated by calculating the energy required for macromolecular synthesis or obtaining this data from studies of related organisms (59; 24).

After formulating the biomass equation, the network can be converted into a stoichiometric model. This process is accomplished through mathematical representations. Here, a steady state is assumed - $equation\ 5$ - meaning the metabolite concentrations are constant and the consumption rates are equal to the production rates of each metabolite.

$$equation\ 5:$$

$$S \cdot v = 0,$$

where S represents the stoichiometric matrix (rows for metabolites and columns for reactions) and v is the flux vector.

In this context, the number of reactions is typically greater than the number of metabolites, meaning that the system is indeterminate. To overcome this problem, it is necessary to reduce the set of possible solutions, adding new constraints to the system (45). Constraints can be categorised as physico-chemical, regulatory, topological and environmental conditions (60). Physiochemical and environmental constraints

are the most used for GEMs (61; 62). The constraints, established within the network, define the minimum and maximum allowable flux for each reaction. This process sets bounds and contributes to a more defined and realistic model.

Model validation

At this final phase, the model must be validated through comparison of *in silico* simulations with experimental data. In cases where there is no agreement with experimental data, the steps mentioned above should be revised, especially those with manual curations.

While the ultimate goal is to represent the totality of the metabolic reality for a given organism in GEMs, it's crucial to recognise the inherent challenges, especially for more complex organisms. Achieving an exact representation of the entire metabolism may be challenging due to its complexity. Nevertheless, the aspiration for very complex organisms remains to provide a comprehensive and accurate portrayal of the organism's metabolic behaviour under specific environmental conditions (63).

In order to ensure a robust and reliable model, biomass precursor production should be the main target when simulating, and, at the same time, it is necessary to identify the missing metabolic functions. The models developed until the moment, have had few rounds of refinement, so a well-employed validation will probably identify inaccurate behaviours of our model that need to be rectified (64). As this process is still a trial-and-error, in which the improvement of the developing model relies heavily on the knowledge of its modeller (64), evaluations supported by simulations are very relevant, as it allows the model to be improved by measuring its accuracy (25).

One way to validate a GEM is through the use of constraint-based methods, such as FBA (65). FBA is a popular method that uses linear programming to identify the flux distributions that maintain the network in a steady state (32). The FBA objective function is usually set to maximise the biomass reaction flux, as this is often considered to be the biological objective of the cell (57). However, this objective function may not be appropriate for all organisms, particularly for multicellular organisms that have different cell types with specific objectives and functions.

Additionally, Parsimonious enzyme usage FBA (pFBA) (66) is an improved approach that considers the selection of strains that minimise the production of enzymes and as a result, require the lowest overall flux in the metabolic network. The Flux Variability Analysis (FVA) (67) further supports the suppression of non-functional metabolic reactions, it usually is used to find the minimum and maximum flux for reactions in the network (68). There are other algorithms that can be used to predict phenotypes with different approaches, but FBA is the most widely used (32).

Some of these methods can be employed to assist in the processes of gap-finding, leading to the identification of new metabolic reactions and functions. However, the simultaneous identification of "dead-end" metabolites may require additional methods or a combination of approaches (7).

These simulations can be performed using tools like COBRApy (29) or OptFlux (69). If there is no agreement with experimental data, the steps mentioned above should be revised. The model can be saved in several different formats (e.g., .mat, .xlsx) but the Systems Biology Markup Language (SBML) format is the standard file format for metabolic models (70). The SBML format ensures compatibility and interoperability across different software platforms and tools, making it a preferred choice for researchers in the field of systems biology (70).

2.2.2 Automatic Reconstruction of GEMs

The automatic reconstruction of GEMs is a powerful approach to the development of these. The main reason for making these procedures automatic is to make the reconstruction much quicker, as opposed to the manual one which consumes considerable time (21; 24). Another reason for the development of automated GEM reconstruction methods is the availability of biological Big Data (71).

Furthermore, automatic GEM reconstruction can also be used to generate models for a large number of organisms simultaneously, enabling large-scale analyses such as the reconstruction of models for the entire human gut microbiome (72). This can provide valuable insights into the metabolic interactions between different organisms within a community and can be used to identify potential targets for biotechnology applications (73).

As mentioned, the traditional manual reconstruction of GEMs is a time-consuming procedure that requires a large amount of data to be examined. To overcome this challenge, several software programs have been developed for automatic GEM reconstruction - these tools automate various aspects of GEM reconstruction - including the annotation of the genome sequence, the generation of GPR associations, the prediction of reaction reversibility, and enzyme localization (74).

One of the early methods for automated GEM reconstruction was GEM System (75), which assigns functions to genes in a target genome through homology and orthology searches against protein databases and maps appropriate reactions to metabolic genes based on EC number matches to the KEGG (47) pathway databases (76). AUTOGRAPH (77), another early method, uses published models as a template and performs an ortholog search from a target genome to reference genomes to map genes and their GPR associations (76). However, these early methods had limitations, such as producing nonfunctional models that were incapable of simulating biomass production, and required significant additional manual curation

to enable biomass production (76).

More recent and updated methods, such as *merlin* (78; 2), Pathway tools (79), CarveMe (80), ModelSEED (81), and the RAVEN Toolbox (82), have improved upon the capabilities and outputs of early methods. These offer additional features, such as tools for the curation of annotations, sub-cellular localization prediction, GPR generation, graphical interfaces and network refinement and evaluation (76; 83; 84). In Table 1, it is possible to compare the availability of some features for these mentioned recent tools. Although *merlin* (2) has different features, proving to be a robust tool and being widely used, it still does not incorporate gap-filling solutions (2). Regarding the other mentioned tools, ModelSEED (81), integrates the capability to generate draft models, perform automated gap-filling and evaluate the network reconstruction through flux balance analysis and phenotype datasets (84; 76). The KBaseversion of ModelSEED (81) also has the capability for automated reconstruction of multiple models at once, allowing for extensive analyses, such as the building of 8,000 central metabolism models for various microbial genomes. With respect to RAVEN Toolbox (85), it provides methods for network visualisation and analysis in addition to the network reconstruction tools and implements techniques for the prediction of sub-cellular localisation of proteins; it also provides a MATLAB (86) package for easy use. Concerning CarveMe (80), it uses a top-down approach to build single-species and community models in a fast and scalable manner, being able to provide a complete automatic reconstruction of these models (80). The key features of Pathway tools (79; 87) point to its flexibility for data integration, visualisation and analysis of biological systems (87); however, it does not use compartmentalisation information, which could be important for simulating the behaviour of metabolic pathways (88).

Table 1: Availability of important features for automatic reconstruction tools; partially adapted from Capela et al., 2022 (2)

Availability of features for automatic reconstruction tools					
	merlin (2)	Pathway tools (87)	CarveMe (80)	ModelSEED (81)	RAVEN Toolbox (82)
Genome annotation	yes	yes	yes	yes	yes
Compartmentalisation	yes	no	yes	yes	yes
GPR generation	yes	yes	yes	yes	yes
Transporters annotation	yes	yes	yes	yes	no
Gap-filling	no	yes	yes	yes	yes
Pathway visualisation	yes	yes	no	yes	yes
Graphical interface	yes	yes	no	yes	no

Despite the significant advancements in automated GEM reconstruction, there still remains a challenge in evaluating the quality of automatically generated draft GEMs and automating the refinement procedure (89). Draft models made with these tools need to be manually curated and refined (80; 90). This process involves verifying the accuracy of parameters of biomass composition, reaction ratios, metabolite uptake, and gap-filling of metabolic reactions based on experimental data (91; 90). Solutions such as memote (92), a software program that assesses the quality of draft GEMs (21). It allows users to validate and improve their metabolic models by running a series of tests and evaluating the model based on various metrics such as the level of annotation of reactions, metabolites, and the consistency of stoichiometry (92).

Merlin

merlin (2) is a user-friendly Java™ application that automates the reconstruction of genome-scale metabolic models for a wide range of organisms. It is designed to facilitate the transition from genome-scale data to SBML (70) metabolic models, allowing the user to have a preliminary view of the biochemical network (78). The reconstruction process performed by *merlin* (2) includes several steps, such as the functional genomic annotations of the whole genome, the identification and annotation of genes encoding transport proteins, the generation of transport reactions and the compartmentalisation of the model.

One of the key features of this tool is its ability to perform semi-automatic enzymatic (re-)annotation

of an organism's genome (78). *merlin* (2) also includes tools for the identification and annotation of genes encoding transport proteins, as well as the generation of transport reactions for such carriers (93). Additionally, it allows loading compartmentalisation information provided by tools, like PSORTb 3.0 (94), LocTree3 (53) and WolfPSORT (54) (78).

All of these operations, together with a unique tool for determining GPR associations, allow performing the main tasks required to obtain reliable models (2). The output of these tools is a set of reactions with GPR rules, which can be edited and viewed by the user. Finally, *merlin* (2) allows exporting the model in SBML (70) format, with MIRIAM (95) annotations, to make it easier for other applications to compare, combine, and reuse the metabolic models (78). However, *merlin* (2) does not have a built-in gap-filling feature. Nevertheless, it is possible to use other software tools in conjunction with it to perform gap-filling on the models it generates. In the sense of using exclusively this tool, it would be necessary to manually curate the metabolic network to fill the gaps in the network.

2.2.3 Gap-filling of metabolic models

A draft model, generated through automatic reconstructions, contains knowledge gaps (24; 96). Usually, it is required the usage of algorithms to fill these gaps (1). This process converts the draft model into an actual functional one. Orth and Palsson (2010) (97) already reviewed the classical algorithms for this, but a few descriptions of these will be given further. The concept of gap-filling appeared with the first GEM reconstructions (98). It involves adding missing metabolic reactions to an incomplete network, improving its connectivity and allowing for more accurate refinement and evaluation (4). Furthermore, it becomes possible, after a transformation of a draft model, to simulate biomass production for a specific growth medium (99; 1). Using reference databases with information regarding metabolic reactions, it is possible to deliberate additions of reactions to the network (1).

A gap-filling analysis leads to the inclusion of new reactions, unknown pathways, unannotated and misannotated genes, as well as promiscuous enzymes and underground metabolic pathways (4). With the increasing availability of high-throughput data for many organisms, gap-filling methods are likely to lead to many discoveries in the future.

When adding missing metabolic functions to an incomplete GEM, it may create new gaps or allow the model to perform functions that the target organism is unable to. For this reason, being prudent is necessary at this stage. However, in some cases, gap-filling is necessary to guarantee the functionality of the model, such as the synthesis of biomass precursors. It is generally recommended that a gap reaction should not be added to the model if there is no information supporting its existence unless it is

necessary for the model's functionality. Manual curation of the results of gap-filling is usually needed to obtain high-quality models (10).

There are many premises to address the gap-filling problem. Usually, it can be faced through the network topology and genomic data, occupied by finding genes related to the missing reactions (9). Furthermore, there is the requirement to minimise the difference between the model and experimental data (9). Several methods were designed to tease out missing reactions within the draft models. Some of the most popular algorithms are described further, however, these can be, generally, described in three main steps (4; 9):

- Detecting gaps by identifying dead-end metabolites that cannot be produced or consumed, as well as any discrepancies in the predictions of the draft model and experimental data;
- Suggesting changes to the model content by adding a set of reactions to fix the problems from the previous step;
- Identifying the genes responsible for the gap-filled reactions.

Problem of gap-filling in automatic reconstructions

One of the main problems with automated gap-filling in metabolic network reconstruction tools is the potential for adding false positive reactions (4). This can occur when the gap-filling algorithm adds a reaction to the model that is not available in the actual metabolic network of the organism, which may lead to incorrect predictions and can impair the overall accuracy and utility of the metabolic model (4). Another problem relies on gap-filling algorithms adding reactions without robust genomics evidence to the models. This can allow the model to perform a biological function that the organism is not able to do (4). To assess the accuracy of gap-filling algorithms, benchmark tests are often conducted. These tests provide a way to assess the accuracy and effectiveness of these algorithms by comparing the results of the gap-filled metabolic network against experimental data (9).

Additionally, gap-filling methods are often computationally intensive, which can limit their applicability and make them less practical to use on a large scale (32).

Gap-filling algorithms

Many tools have been developed to proceed with gap-filling on the metabolic networks (1), some of them and their characterisation were summarised in Table 2. A few of these will be described in this section, starting with the classical methods, and then the more recent and novel ones will also be slightly described.

The classic algorithms for this purpose and orphan-filling include SMILEY (100), SEED (101; 102) and ADOMETA (103; 104; 105). Regarding SMILEY (100), the first algorithm made for this intent, it uses linear programming, identifying the minimum number of reactions needed to be added to a metabolic model from a database of reactions, enabling a minimum defined growth rate to be achieved (106). Concerning SEED (101; 102), it uses a subsystems approach and comparative genomics to find candidate genes for orphan reactions. The last one to be mentioned is ADOMETA (103; 104; 105), which combines different types of functional association evidence, such as gene co-expression and phylogenetic profiles, to identify potential enzymes for orphan reactions (103; 104; 105).

Typically, algorithms for gap-filling or for filling dead-end reactions may utilise approaches based on network topology (84) but also tackle the problem with other sources of information, using pre-defined pathways (79) or phenotype data (107; 108; 109; 110). Over time, new algorithms were developed aiming to increase the efficiency of detecting and filling the gaps; some of these are now being briefly described. As such, FastGapFill (111) is a scalable algorithm that computes a near-minimal set of added reactions for a model. For this algorithm, gap-filling is performed by optimising an objective function that minimises the number of added reactions while ensuring that the model remains consistent with the constraints (4). Another method, GLOBALFIT (112), identifies the minimal set of changes needed to correctly predict experimental growth and non-growth, finds a globally optimal network, and considers all experiments and all possible changes simultaneously. A novel tool, Meneco (1), solves this problem using Answer Set Programming (ASP) (113) and considers reactions as achievable only if their reactants are available. As this tool is considered very relevant and uses topological analysis, other aspects regarding its functioning are described in the next section. As an example of new algorithms that utilise alternative mechanisms to find missing reactions emerges GAUGE (3). It uses Flux Coupling Analysis (FCA) (114) to determine the relationships between metabolic genes and mixes it with a step, where the metabolic network is reconstructed by adding a minimum number of reactions that minimise the discrepancy between the experimental data and the predicted flux coupling relations (3; 4). Important to mention, BoostGAPFILL (9), combines topology and constraint-based approaches to make predictions about missing reactions in a metabolic network. The method uses matrix factorization models to complete a partial adjacency matrix derived from the incomplete stoichiometric matrix, then formulates the selection of reactions from a universal database as an integer least squares problem (9).

Table 2: Tools for gap analysis and their different approaches; partially adapted from Hosseini and Marashi, 2017 (3)

Tools for gap-filling and their characterisation

Tool	Approach for model inconsistency	Optimisation algorithm	Strategy
GapFill (84)	Topological	MILP[a]	Minimising added reactions
FastGapFill (111)	Topological	LP[b], MILP	Minimising added reactions
SIMLEY (100)	Growth phenotype data	MILP	Minimising added reactions
FastGapFilling (115)	Growth capability	Heuristic, using LP	Maximising biomass flux, Minimising added reactions
Meneco (1)	Topological	ASP[c]	Minimising added reactions
GAUGE (3)	Flux coupling analysis	MILP	Minimising added reactions
minimalExtension (116)	Converting nutrients to target metabolites	Greedy	Minimising added reactions
BoostGAPFILL (9)	Topological, constraint-based	ILS[d]	Minimising added reactions

[a] Mixed integer linear programming; [b] Linear programming; [c] Answer set programming; [d] Integer least squares

2.2.4 Tools for Topological Gap-filling

Using constraint-based methods for gap-filling procedures may fall short in capturing the topological information of a given metabolic network (9). Topological characterisation of a metabolic network employs edges to represent reactions and nodes to represent metabolites (117). This representation allows for the analysis of metabolite producibility, dependent on the existence of a path in the network from precursor metabolites to the target metabolite.

The producibility of a given metabolite refers to its capacity for synthesis from a set of precursor metabolites through a series of metabolic reactions in the network. This concept is based on the presence of a path in the network connecting precursor metabolites to a target metabolite, relying on enzymes catalysing reactions along the path and the availability of precursor metabolites. Another method to determine producibility involves stoichiometry, where linear constraints on fluxes through network reactions theoretically enable metabolite production, concluded through linear programming and optimisation techniques (84).

When applied to gap-filling, the topological approach may be more robust and reliable than stoichiometric-

based gap-filling (1). This is because it does not rely on specific concentrations of metabolites but rather on the presence or absence of reactions and their connections in the network, offering greater resistance to errors.

There are methods for gap-filling that purely rely on the topological aspect of metabolic networks and do not require additional information. Classical methods reconstruct the network based on flux consistency, namely GapFill (84) and FastGapFill (111). However, relying solely on topology may not guarantee the resolution of dead-end metabolites.

Besides, other methods, such as BoostGAPFILL and Meneco (9; 1), consider the topological characterisation of the metabolic network. Notably, Meneco will be emphasised due to its consideration of topological features and, with its synergy with BioISO, the capacity to handle dead-end metabolites.

2.2.5 Meneco

Meneco is a computational tool designed to address the complex challenge of gap-filling in metabolic networks, particularly those derived from recent investigations of complex organisms (1). Its distinctiveness relies on its ability to identify essential reactions in metabolic networks, especially when traditional stoichiometric-based methods might falter.

Objective and Optimisation

In metabolic network reconstruction, the goal is to ensure that a set of target metabolites, denoted as M, can be synthesized from a set of available reactions, R. The producibility of a metabolite m from M within the network can be represented as a binary function:

$$P(m) = \begin{cases} 1 & \text{if } m \text{ is producible from } R \\ 0 & \text{otherwise} \end{cases}$$

The objective of gap-filling is often to identify a minimal set of reactions, ΔR, from a reference database, such as MetaCyc (118) or KEGG (47), to ensure all metabolites in M become producible. This optimisation problem can be formulated as:

$$\text{Minimise } |\Delta R|$$

$$\text{Subject to } P(M \cup \Delta M) = 1$$

Here, ΔM represents the set of previously non-producible metabolites, and the goal is to minimize the number of reactions added from the reference database.

Answer Set Programming

ASP stands as a robust declarative and non-monotonic logic programming paradigm, wielding significant prowess in addressing intricate combinatorial challenges. ASP introduces a systematic and automated framework for deriving solutions, herein termed "answer sets," that conform to meticulously defined logical rules and constraints. Its application spans diverse domains, including artificial intelligence, knowledge representation, and the burgeoning field of computational biology.

ASP embarks on the delineation of a given problem by articulating a set of logical axioms and constraints using a formally prescribed language. These axioms encapsulate the underlying logic, constraints, and overarching objectives intrinsic to the problem at hand. The overarching aim is to ascertain one or more answer sets, guided by the optimization of specific criteria.

Central tenets characterising ASP encompass:

- **Declarative Nature**: ASP affords problem solvers a lucid and concise avenue to articulate the requisites and constraints intrinsic to the problem statement. This declarative prowess positions it as an apt choice for tackling intricate problems characterized by intricate logical interplay.
- **Non-Monotonic Reasoning**: ASP excels in the realm of non-monotonic reasoning, rendering it capable of accommodating scenarios where fresh information may necessitate revisions or augmentations within the purview of answer sets. This adaptive trait becomes paramount when dealing with dynamic and evolving problem landscapes.
- **Automated Solvers**: An array of efficient ASP solvers has arisen, dedicated to the automated derivation of answer sets for a given problem instance. These solvers deploy sophisticated algorithms to expedite the quest for solutions in an efficient manner.

Within the context of the Meneco (1) algorithm, ASP assumes a vital role, predominantly in the usage of the `get_minimal_completion_size` method. This method harnesses the capabilities of ASP to delineate a subset $R' \subseteq R$ that aligns with stringent criteria for the producibility of target metabolites. ASP's intrinsic capacity to handle intricate logical reasoning and optimisation renders it an apt choice for this endeavour.

The `get_minimal_completion_size` method, ensconced within Meneco, orchestrates its mission with dual foci:

1. **Producibility Mandate**: For each target metabolite t extant within T, the method endeavors to ensure the producibility of t via the amalgamation of reactions from D and R', accompanied by the

seed metabolites S. This undertaking necessitates intricate logical scrutiny, unravelling the enigma of optimal reaction combinations conducive to the producibility of all target metabolites.

2. **Minimisation Directive**: Concurrently, the method undertakes the task of minimising the cardinality of R' ($|R'|$), driven by the aspiration to discern the most parsimonious set of reactions requisite for the fulfilment of producibility prerequisites. This optimisation endeavour mandates the curtailment of reaction count while steadfastly preserving the desired producibility outcomes.

However, this method only provides one of several possible solutions with the same size. This package also includes methods to get all possible solutions for a given number of reactions.

Topological Approach

Meneco's approach diverges from traditional methods by focusing on the network's topology. For a given metabolite m, its topological producibility is determined by the paths in the network leading to m. If no such path exists, m is deemed non-producible. This topological producibility can be represented as a matrix T where each entry T_{ij} indicates the producibility of metabolite i using reaction j.

Draft Network and Gap-Filling

One of the primary challenges in metabolic network reconstruction is the presence of gaps, which are reactions missing from the network that prevent the production of certain metabolites. Meneco addresses this challenge by leveraging a draft network and a database of biochemical reactions to identify and fill these gaps (1). Importantly, the tool ensures that the added reactions are biologically relevant and do not introduce inconsistencies into the network.

Versatility and Systems Ecology

Meneco's topological approach allows it to operate without the need for a biomass function or growth simulations, making it suitable for networks of non-model organisms or those derived from metagenomics data where such information might be lacking (1). Additionally, it has broader implications in systems ecology. For instance, in a symbiotic relationship between two organisms with metabolic networks R_1 and R_2, the combined producibility can be represented as:

$$P_{combined}(M) = P_{R_1}(M) \cup P_{R_2}(M)$$

This represents the union of producible metabolites from both organisms, highlighting potential metabolic complementarities in symbiotic relationships.

Known limitations of Meneco

While this is considered a powerful tool for metabolic network completion, it does present certain challenges and limitations, especially when applied to organisms distantly related to common model organisms. One of the primary challenges is the determination of the biomass reaction. Often borrowed from well-established model organisms, this reaction might not encapsulate the unique characteristics of the studied organism, particularly when dealing with extremophiles, where metabolic pathways may significantly differ (119).

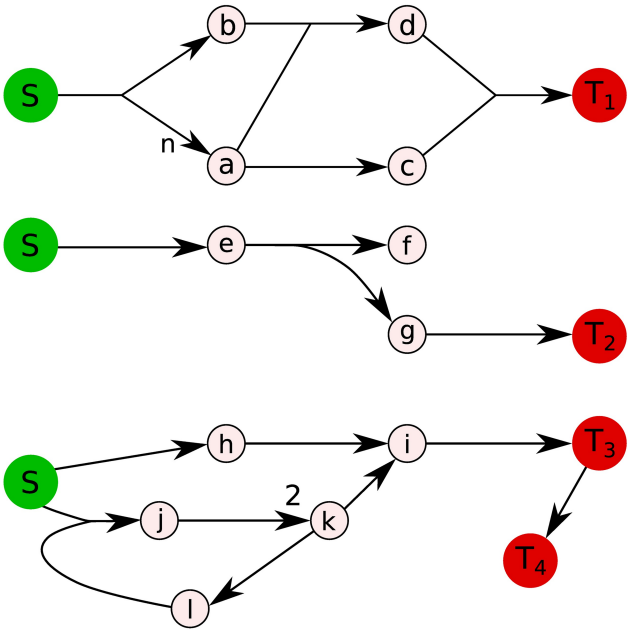
Furthermore, the identification of boundary compounds, dead-end metabolites (targets), and cofactors becomes intricate. These elements might be challenging to characterise solely from experiments or existing literature. This poses challenges, especially when considering the stoichiometric balance of metabolic reactions. While score-based methods are commonly employed, they may be susceptible to errors, particularly in cases where the stoichiometry is complex or not well-defined (120).

Meneco's researchers acknowledge the potential for false positives in examples that generated hypotheses for algal-bacterial interactions. Ongoing experiments aim to validate these findings; however, 58 targets (70%) resulted in false positive outcomes. This underscores challenges such as missing gene annotations or erroneous assignments. Importantly, these difficulties underscore the necessity for cautious interpretation and ongoing experimental validation to enhance the accuracy of predicted algal-bacterial interactions.

The need for gap-filling techniques tailored for newly developed model organisms is evident, especially for those with intricate evolutionary histories or those existing in extreme environments where phenotypic data might be sparse (121).

Meneco's approach to gap-filling, as described by Prigent et al. (1), is qualitative and combinatorial. While it efficiently determines the bio-synthetic capacities of metabolic networks based on their topology, it might not always capture the intricate complexities of metabolic interactions. This becomes evident when compared with methods that offer a more integrative or quantitative perspective. This tool uses qualitative constraints to express the producibility of metabolites based solely on the topology of the metabolic network. However, this topological approach can sometimes overlook certain nuances. For example, databases often do not precisely describe the species-specificity of cofactors in reactions or the stoichiometry of reactions. Such omissions can lead to prediction errors for degraded metabolic networks when producing

biomass (1). Furthermore, Meneco (1) defines the synthetic capability of a system concerning a set of input compounds but excludes the self-production of a compound via cycles. This exclusion contrasts with mass-balanced stoichiometric frameworks that allow such cycles. This difference can lead to discrepancies in predictions when juxtaposed with methods that consider these cycles. Figure 1 illustrates the differences between topology and stoichiometry-based producibility of compounds in a metabolic network. The Figure is divided into three parts, each highlighting a specific aspect:



Compound	Topological producibility (Meneco)			Stoichiometric based producibility (FBA)		
	n=1	n=2	n⩾2	n=1	n=2	n⩾2
S	✓					
a b c d	✓			✘	✓	✘
T_1	✓			✘	✓	✘
e f g	✓			✘		
T_2	✓			✘		
h i	✓			✓		
j k l	✘			✓		
T_3 T_4	✓					

Figure 1: Comparison of topology and stoichiometry-based producibility of compounds in a metabolic network. The figure highlights the differences in producibility criteria between Meneco's topological approach and a stoichiometric framework. Adapted from Prigent et al. (1).

- **Part 1:** This section illustrates the impact of stoichiometric coefficients on different definitions of producibility. While all metabolites would be producible from the seed S using a topological approach, the stoichiometric coefficient n plays a significant role in producibility when using a (FBA)-based approach. For instance, to produce $T1$, an equal quantity of d and c is required. The stoichiometric approach necessitates that n be twice the value of the stoichiometric coefficient of b. Interestingly, if the objective function was solely formed by the reaction producing metabolite d, n would need to be 1 for a stoichiometric perspective, whereas d would always be producible from S in a topological approach.

- **Part 2:** This section demonstrates that while $T2$ can be produced according to graph-based criteria, the accumulation constraint on f blocks its production in a balanced-mass stoichiometric

framework. Conversely, k remains FBA-producible through a cycle involving j, k, and l, but it is not producible according to graph-based criteria.

- **Part 3:** Here, the self-production of a compound via cycles is discussed. In metabolic networks, cycles represent a series of reactions where an initial metabolite is regenerated, allowing for its continuous production. For instance, in the context of Figure 1, the cycle involving j, k, and l exemplifies this concept. Once initiated, such cycles can continuously produce compounds like $T3$ and $T4$. However, Meneco's topological approach does not inherently recognise this self-sustaining nature of cycles. Instead, it requires all inputs of a cycle, such as j, k, and l, to be independently produced to initiate the cycle's reactions. This perspective might overlook the potential of a network to continuously produce compounds like $T3$ and $T4$ if they are part of a self-sustaining cycle. In contrast, mass-balanced stoichiometric frameworks acknowledge the continuous operation of these cycles. As long as the necessary stoichiometric conditions are met, compounds within a cycle, such as $T3$ and $T4$, can be produced indefinitely. This difference in recognising the role of cycles leads to the observed distinction in the producibility of $T3$ and $T4$ between Meneco's approach and stoichiometric frameworks, as depicted in the mentioned Figure.

Moreover, Meneco (1) employs the ASP paradigm to efficiently model the logic of bio-synthetic producibility and solve the gap-filling problem as a combinatorial optimisation problem. However, due to the parsimonious criteria used, Meneco (1) might miss certain reactions, especially those involved in cycles, unless a metabolite from the cycle is added to the seeds. As a graph-based approach, Meneco (1) may not always capture the intrinsic non-linearity of metabolic behaviours and flux imbalances, which are better addressed by tools that consider the stoichiometry of metabolic reactions (1).

2.2.6 BioISO

BioISO is a computational tool designed to facilitate the objective-oriented curation of metabolic networks. Leveraging a recursive relation-like algorithm grounded in FBA, it offers a targeted analysis of metabolic networks, thereby enhancing the efficiency and effectiveness of the curation process. This tool stands as a beacon in the evolving field of systems biology, bringing automation and efficiency to the process of Genome-Scale Metabolic Models reconstruction (11).

User Interface and Accessibility

BioISO offers a user-friendly interface accessible both as a python package, a web service, and a *merlin* plugin, democratising metabolic network analysis for individuals without extensive computational backgrounds. This feature facilitates a deeper understanding of metabolic network structures and functions (11).

Objective-Oriented Approach in Metabolic Network Curation

Central to BioISO's functionalities is the objective-oriented approach, which facilitates targeted and efficient analysis of metabolic networks. This approach, grounded in the meticulous identification and analysis of specific objectives within a metabolic network, aids in understanding the overarching functionality and identifying potential gaps within a metabolic network (11).

BioISO (11) employs a recursive relation-like algorithm deeply anchored in the principles of FBA. This algorithm constructs a hierarchical structure based on the metabolites and reactions associated with a specific objective, offering insights into the metabolic network. The depth of BioISO (11), indicative of the number of recursive calls executed, can be modulated to facilitate shallow, guided, or nearly exhaustive searches, contingent on the metabolic network's size and complexity (11).

Algorithmic Depth and Hierarchical Analysis

BioISO's (11) recursive relation-like algorithm delineates the relationships between different metabolites and reactions associated with a specified objective, constructing a hierarchical structure that visually represents the metabolic network. This detailed visualization aids in identifying potential areas of discontinuity and gaps, pinpointing the precise areas requiring attention during the gap-filling process, thereby advancing the field of systems biology (11).

Identifying Targets and BioMeneco Integration

BioISO (11) excels in identifying targets within metabolic networks, a crucial step in understanding the functionality and potential shortcomings of a metabolic network. This process is central to pinpointing blocked reactions and dead-end metabolites, which are relevant for the development of high-quality GEMs (11).

BioMeneco (11), a development that integrates BioISO (11) with Meneco (1), was conceived to explore whether BioISO could enhance Meneco's results by narrowing down the search space during the gap-filling task. This integration leverages the power of answer set programming to find the most efficient

completions of draft genome-scale metabolic networks, thus facilitating a more streamlined and efficient gap-filling process. It aids researchers in constructing more accurate and complete metabolic networks, enhancing the efficiency and effectiveness of the metabolic network reconstruction process (11).

During the validation process, specific reactions were removed from two different models to test the system's efficacy in suggesting potential solutions for restoring the models' prediction of a growth phenotype based on BioISO's suggestions for the set of targets, which served as the primary input for Meneco. The results showcased that BioISO could facilitate high-quality bottom-up reconstructions by adjusting the guided-search gap-filling tool Meneco, proposing BioMeneco as an iterative process comprising two separate tasks: identifying the set of target metabolites not being produced or consumed (dead-end metabolites) using BioISO, and then running Meneco with the previously identified set of metabolites to obtain efficient solutions for completing draft metabolic networks (11).

Chapter 3

Materials and Methods

As previously emphasised in the State-of-the-art chapter, understanding the representation of a given network is of great importance. The metabolic model, hypothetically denoted as M, is typically represented as a graph. In this representation, nodes symbolise metabolites while directed edges depict reactions. This structure allows for the application of graph-based analyses such as shortest path or connectivity assessments, thereby aiding in the exploration and analysis of metabolic pathways.

Grasping the computational representation of metabolic models is central to this study. It not only fosters a detailed analysis of metabolic pathways but also lays the groundwork for easily developing consistent and efficient analyses of the metabolic networks. To facilitate this representation and further manipulations, several modules were developed. Among them, the COBRApy module (29) was extensively utilised for its robust capabilities in handling and analysing metabolic models.

This chapter delineates the materials and methods utilised in this study, highlighting the tools, methods, and modules that were indispensable in developing techniques aligned with the objectives of this thesis.

Further sections in this chapter will delve into the specifics of the methodologies employed, the computational tools utilised, and the analyses performed to achieve the objectives of this thesis.

3.1 Development of a Universal Model

The gap-filling with Meneco requires a database with reactions and metabolites in SBML format. This often includes data from one or more biological databases, like MetaCyc or KEGG. For this work, the KEGG database was used. This database was retrieved and filtered to remove undesired reactions and compounds, like DNA biosynthesis reactions and glycans. Additionally, incomplete reactions were also removed as they would introduce errors in the gap-filling. For instance, the reaction *R08585* converts chlorophyll a directly into chlorophyll b, ignoring the requirements for NADPH, oxygen, and phytyl diphos-

phate. In total, 12 reactions were removed from the universal model. As KEGG reactions are all reversible by default, the "Correct reversibility" tool, available in *merlin* was used selecting ModelSEED as an information source.

3.2 Developed Workflow

This section elucidates the workflow engineered for the integration of methods used in the gap-filling process of genome-scale metabolic models. A significant feature of this workflow is the conjugation of Meneco and BioISO methods, requiring an initial bridging of these frameworks followed by the formulation of a novel set of methods to optimise the process. The primary aim of this workflow is to streamline the gap-filling process, with a focus on enhancing computational efficiency and generating viable solutions that identify a set of reactions capable of tackling the existing gaps.

The workflow requires the submission of a draft model by the user, as well as a configuration file. The generation of a draft model for a specific organism can be accomplished with GEM reconstruction tools, like *merlin*.

The workflow is structured to include various methods ensuring the derivation of optimal solutions. It is organised into the following main phases 2

- Identification of seeds and targets: This phase involves identifying the seeds and target metabolites which are crucial for the subsequent gap-filling process. The BioISO module is here utilised to detect the targets.
- Replicating all reactions and metabolites in the universal model for each compartment of the draft model
- Building a custom universal model: A custom universal model is created to encompass a specific range of metabolic reactions and pathways related to the draft model.
- Evaluate reconstructable and unreconstructable Targets, and identify requirements for additional seeds
- Running Meneco's algorithm: Meneco's algorithm is employed to suggest a set of reactions to fill these gaps, thereby enhancing the completeness and accuracy of the model.
- Evaluating Meneco's results with FBA and generate report

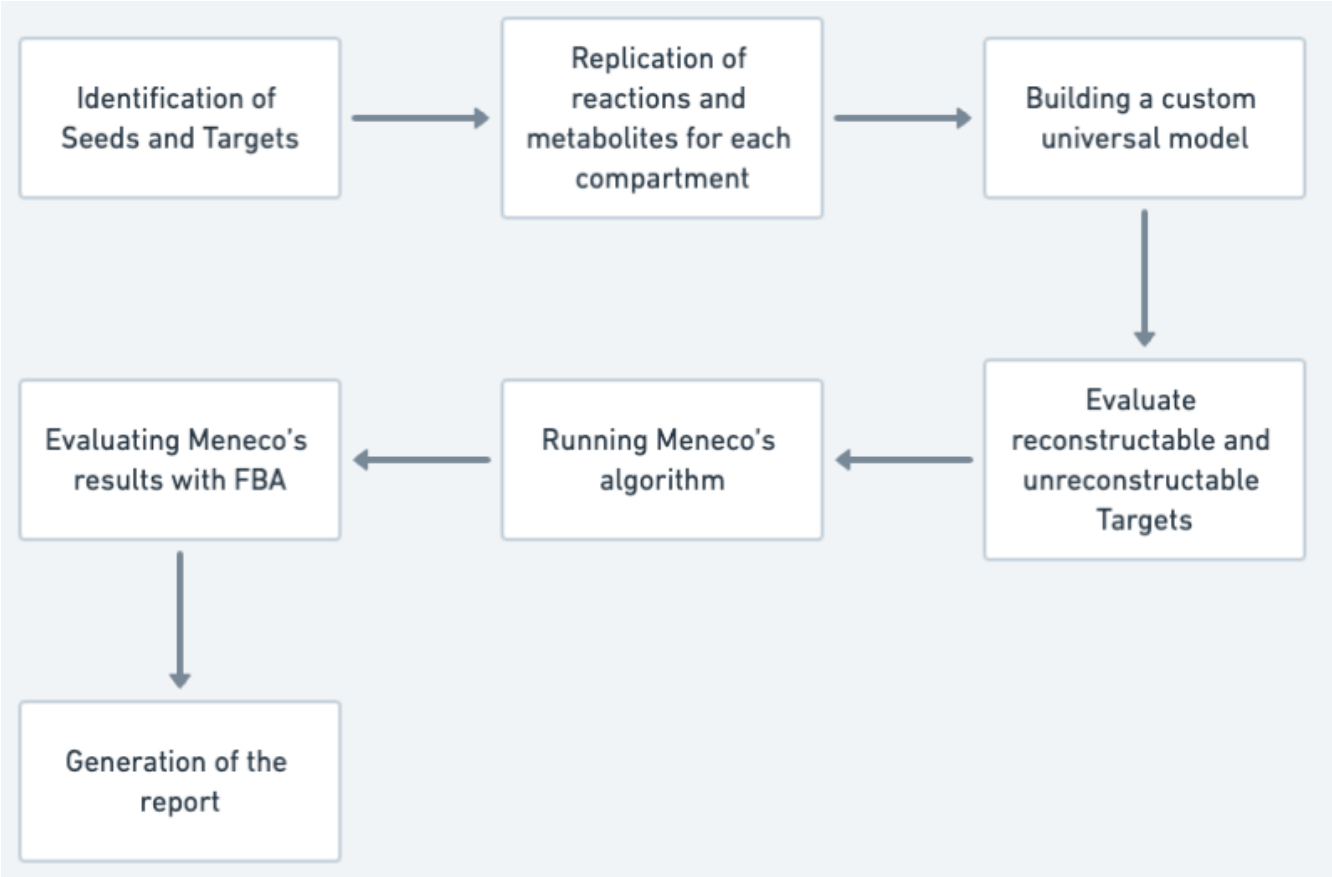


Figure 2: Diagram illustrating an overview of the main phases of the Developed Workflow

3.2.1 User inputs

The workflow developed in this work requires two inputs from the user: a draft metabolic model in SBML format, and a configuration file in JavaScript Object Notation (JSON) format (122). Figure 3 presents a diagram with all major steps of the workflow. The subsequent sections detail the specificities of each step.

The draft model submitted by the user must be in SBML format. The model is read by COBRApy (29) to allow easy reading, writing, and manipulation of the model. This approach not only enables users to tailor the model's parameters to their specific research needs without altering the core codebase but also ensures that analyses can be easily shared, replicated, or modified under different conditions. The simplicity of parameter specification is further enhanced by the configuration file.

For instance, consider the structure of the user-defined JSON file. An example of this file can be accessed at the following link: ExampleSubmissionParameters.JSON. This file contains key parameters that dictate how the metabolic model will be processed and analysed. The `objective_function_id` (mandatory parameter) specifies the objective function of the metabolic model that the user intends to refine through gap-filling (e.g., `e_Biomass__cytop`). This is crucial as it defines the primary metabolic

activity or goal that the model aims to achieve, in this case, biomass production.

The `sinks` (optional parameter) section lists metabolites, which will be used as seeds. For each metabolite listed here, a sink reaction will be generated. In this example, it includes `C00002`, which represents ATP, for three different compartments, as well as glucose (`C00031`) in the cytoplasm. This allows the user to easily add seeds to the model.

The `compartments` (optional) parameter enumerates the different cellular compartments present in the model. It defines which compartments will be replicated in the universal model. This can be useful if the user wants to do gap-filling in one particular compartment, or if it wants to ignore some of them (e.g., extracellular environment). Thus, This is essential for compartmentalised models where reactions and metabolites are segregated into distinct cellular locations. In the provided JSON example file, these compartments include the cytoplasm (`cytop`), peroxisome (`pero`), chloroplast (`chlo`), mitochondria (`mito`), endoplasmic reticulum (`er`), and Golgi apparatus (`golg`). By introducing this mechanism for user-defined parameters, the methodology is transformed into a versatile instrument.

Another relevant input parameter is `max_solutions` (optional). This parameter delineates the maximum number of solution sets the algorithm should consider when identifying potential reactions to tackle the gaps in the metabolic network. The rationale behind exploring multiple solutions is to mitigate the inherent risk of relying solely on the minimal solution. While the minimal solution provides an efficient set of reactions to produce the targets, it doesn't guarantee success in every scenario. By examining multiple solutions, the methodology increases the likelihood of identifying a set of reactions that successfully enable biomass production.

3.2.2 Seeds and Targets identification

The identification of targets and seeds is crucial both to the results of the workflow and the required running time. To perform such identification, the class Model was created. The source code of this implementation can be found here.

Creating the Model instance

Although COBRApy provides several methods useful for manipulating metabolic models, the workflow demands creating additional methods specific to the workflow. Thus, the Model class was created, inheriting the `cobra.Model` class. The instances of this class act as the central entities for subsequent manipulations and in-depth analyses of metabolic models. It not only encapsulates the entire metabolic network but also offers a suite of methods that facilitate querying, modifications, and the identification of relevant

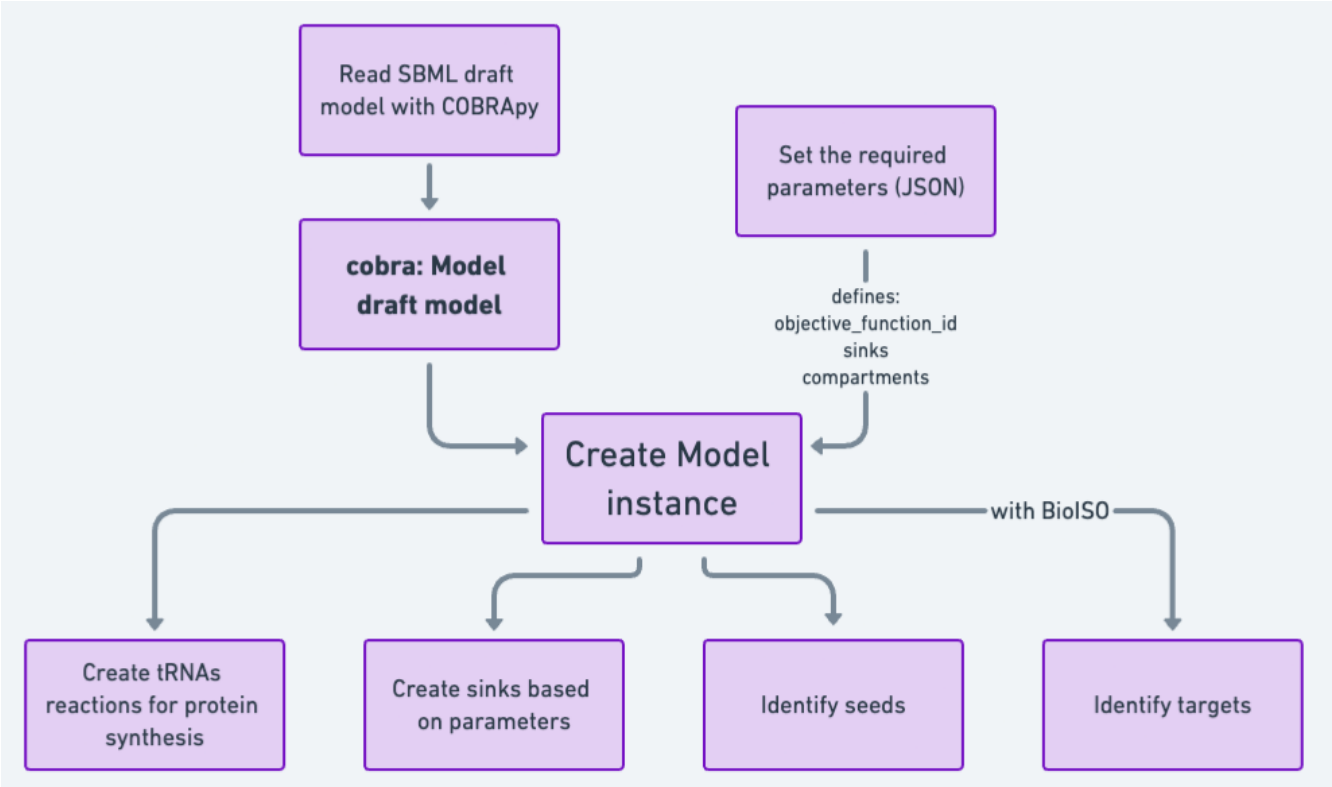


Figure 3: Diagram illustrating the Model instance creation process

components for further gap-filling analysis and procedures. For a deeper dive into the source code, one can refer to the `model.py` file available on the GitHub repository: link.

The instantiation process primarily requires two parameters: a "cobra" model and the previously discussed *objective_function_id*. During the initialisation phase, the object constructs both a reaction pathway map and a metabolite pathway map. These maps are then integrated as attributes of the object, providing a structured representation of the metabolic pathways and their interrelationships. This structured representation is invaluable for subsequent analyses, ensuring that the relationships between reactions, metabolites, and their associated pathways are easily accessible and well-defined.

Creation of sinks

Sinks are abstract reactions added to the model that allow for the removal or addition of specific metabolites. They are essential for simulating open systems where certain compounds can freely enter or leave.

In the realm of metabolic modelling, the concept of an open system is crucial. In real-world biological scenarios, organisms do not function in isolation. Certain metabolites can traverse cellular boundaries, freely entering or exiting the system. To computationally capture this dynamic, emerges the concept of sink reactions. These reactions, while not representing actual biological processes, serve as computational

constructs that allow specific metabolites to be added or removed from the system without any constraints.

At the `Model` class, a method `create_sink`, is used to generate such sink reactions. It accepts a metabolite's identifier and, optionally, the rate limits for the sink reaction. By default, these bounds are expansive, set to (-10000, 10000), ensuring a substantial potential for both influx and efflux of the metabolite. The function constructs a reaction name by prefixing the metabolite ID with "Sk_". This nomenclature ensures easy identification of sink reactions within the model. Subsequently, a new reaction is instantiated involving the specified metabolite with a stoichiometry of -1, indicating the potential removal of the metabolite. However, given the bi-directional nature of the bounds, the reaction also permits the addition of the metabolite. Once established, the function assigns the specified bounds to the sink reaction and returns it.

Creation of tRNA reactions

tRNAs are central to the process of protein synthesis, serving as the link between the mRNA sequence and the corresponding amino acid sequence during translation (123). However, due to the approach used to include protein biosynthesis in metabolic models, it is necessary to include sink reactions for tRNAs. The method `create_trnas_reactions` at this file is designed to solve this problem, by checking if the precursors for protein synthesis have been identified. If not, it calls the `get_pre_precursors` method to retrieve them. Following this, it obtains the products associated with the precursor reactions for protein synthesis using the `get_products` method.

This way it iterates through these products, identifying those that represent tRNAs - filtering out unrelated entities like water and the generic protein placeholder ("e-Protein"). For each tRNA identified, a sink reaction is created within the model using the `create_sink` method. This step ensures that each tRNA molecule is accounted for in the model.

Identification of Seeds

In metabolic networks, seeds, are foundational compounds that an organism can assimilate from its environment. These compounds initiate various metabolic pathways, enabling the organism to synthesise a diverse range of metabolites vital for its growth and sustenance. Recognising these seeds is pivotal as they are fed into the gap-filling algorithm, Meneco (1), ensuring the algorithm is equipped with the requisite information for precise prediction and gap-filling in the metabolic network.

The `identify_seeds` method, illustrated below, is crafted to pinpoint these seeds within the model. The procedure entails iterating over the model's boundary reactions, which typically signify exchanges

between the organism and its surroundings. For each boundary reaction, the method scrutinises its reactants. If a reactant hasn't been previously identified as a seed and the reaction's lower bound is negative (indicating potential uptake of the reactant), it is appended to the seeds list. The method culminates by returning a list of tuples, each containing the ID and compartment of a seed metabolite.

```
# method to identify the seeds of a model
def identify_seeds(self) -> List[Tuple[str, str]]:
    total_seeds = []
    total_seeds_ids = []
    for reaction in self.boundary:
        for reactant in reaction.reactants:
            if reactant.id not in total_seeds_ids and reaction.lower_bound < 0:
                total_seeds.append((reactant.id, reactant.compartment))
                total_seeds_ids.append(reactant.id)
    return total_seeds
```

As highlighted in the Meneco (1) paper, there's a recognised advantage in curating a seeds file that's augmented with additional seeds. In response to this, a new method concerning the identification of more specific seeds for each identified target will be presented later in this document.

Targets Identification

In the workflow context, targets are compounds that are supposed to be produced by the organism (e.g., biomass components), but the metabolic network is not able to synthesise them. Identifying these targets is essential as they can indicate incomplete or improperly annotated pathways and their presence can affect the predictive accuracy of the metabolic model.

To discern these targets within the model, the BioISO algorithm was employed. A method, `identify_targets`, was developed to automate this process and integrate it into the workflow. While there are some relevant code snippets presented below, for a comprehensive understanding, it is recommended referring to the full `model.py` file in our GitHub repository: link. Additionally, the BioISO repository can be accessed at: link.

The method starts by setting the objective direction and the solver for the model. The objective can be to either 'maximize' or 'minimize' the flux of a given reaction (in this case, the biomass reaction). The solver can be GNU Linear Programming Kit (GLPK) (124), but it is recommended to use IBM ILOG CPLEX Optimization Studio (CPLEX) (125) as it is more efficient (126). Upon setting these parameters, the method invokes the BioISO algorithm on the model. The algorithm returns a tree structure that represents

the metabolic network, providing a graphical representation of the metabolic pathways and their interconnections. This representation is crucial for understanding the structure of the network and identifying potential dead-ends.

The BioISO instance is created with the objective function ID, the model itself, and the objective (either 'maximize' or 'minimize').

```
bio = BioISO(self.objective_function_id, self, objective)
bio.run(2, False)
```

The `run` method of the BioISO instance is then called to analyse the metabolic network up to two levels deep. The algorithm returns a tree structure that represents the metabolic network. From this tree, the developed method extracts potential target metabolites by accessing the next biomass components from the root of the tree.

```
results = bio.get_tree()
biomass_components = results["M_root_M_root_M_root_product"]["next"]
```

Then, the method iterates over the biomass components, searching for metabolites with the role "reactant". This role is significant as it indicates that the metabolite is not consumed in any subsequent reactions, making it a dead-end or target metabolite.

```
for biomass_component in biomass_components:
    biomass_component_role = biomass_components[biomass_component].get("role")
    ...
```

This loop checks each biomass component to determine if it is a reactant and if it has not been analysed already. If these conditions are met, the component is considered a target.

Once the potential targets are identified, they are saved in a list of tuples. Each tuple consists of the identifier and the compartment of the target metabolite.

```
targets.append((biomass_components[biomass_component].get("identifier"),
                biomass_components[biomass_component].get("compartment")))
```

This structured format ensures that the information about each target is preserved in a concise manner, facilitating subsequent analyses and operations. The `identify_targets` method ultimately returns this list of targets, providing a clear overview of the dead-end metabolites in the metabolic network.

This comprehensive approach ensures a thorough identification of target metabolites, contributing to the next steps of this workflow, in which a targets file will be utilised as input for Meneco's algorithm.

3.2.3 Developing a Custom Universal Model

Creation of the GapFiller instance

In this phase, subsequent to the construction of the seeds and targets models, an instance of `GapFiller` is generated. The complete class can be explored in the `gapfiller.py` file, accessible through the following link.

The `from_folder` method facilitates the automatic construction of this object, necessitating only the folder path where the SBML (127) models are housed and a designated results path. This method conducts a thorough search within the specified folder to locate the essential files (draft model, seeds and targets files) ensuring their presence for the ensuing steps. Following this validation, the method proceeds to read the draft model and commences the cloning of the universal model. During this reading phase, it is crucial to refer to the accompanying diagram (Figure 4), which provides a visual representation of the `GapFiller` instance creation process, offering clarity and aiding in understanding the intricate steps involved.

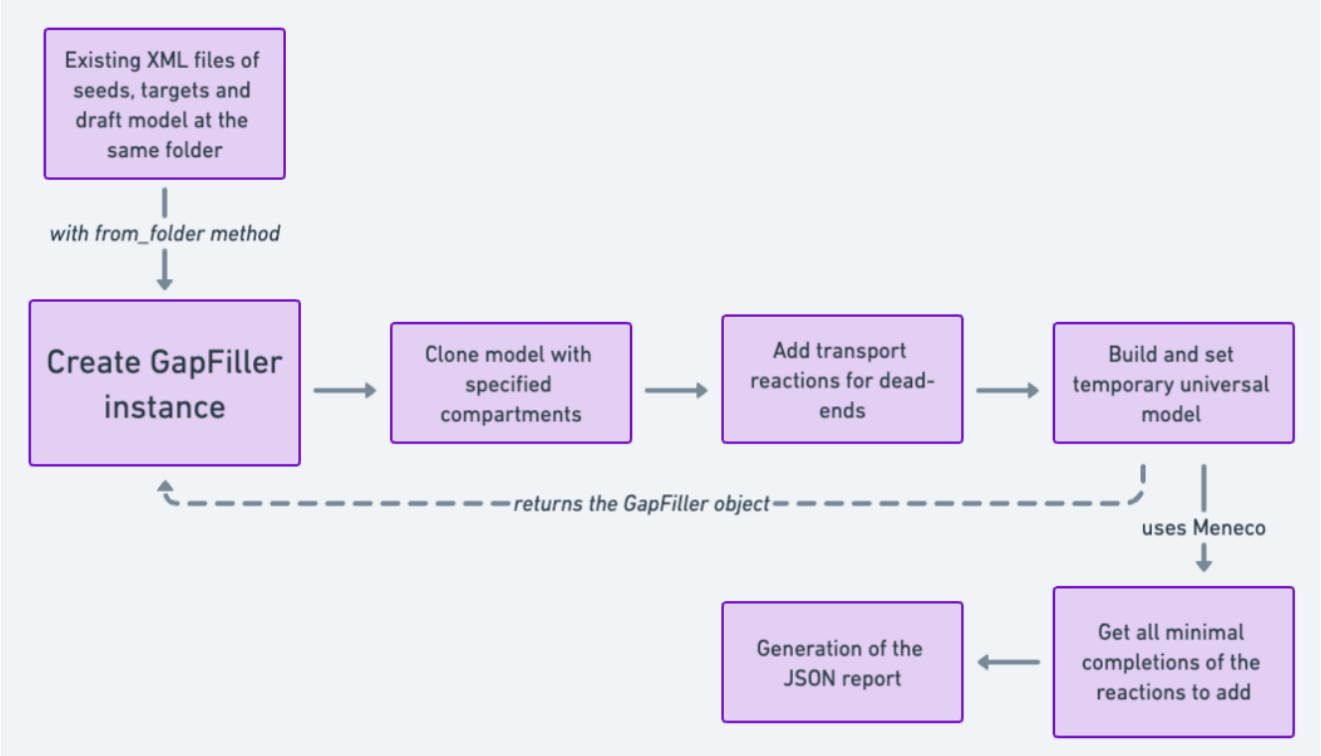


Figure 4: Diagram illustrating the GapFiller instance creation process, including relevant steps such as cloning the model with specified compartments, adding transport reactions, building the Custom Universal Model, utilising Meneco for gap-filling, obtaining solutions, and generating reports.

This structured and systematic approach ensures the seamless creation of the `GapFiller` instance,

laying a solid foundation for the subsequent stages of the metabolic network reconstruction, thereby contributing to the efficiency and reliability of the overall process.

Cloning and Compartmentalisation of the Universal Model

The process of cloning and compartmentalising the model serves as a pivotal foundation, facilitating extensive manipulation and analysis tailored to a hypothetical model under scrutiny. This ensures that the universal model is meticulously adapted to the nuances and prerequisites of the hypothetical metabolic model, paving the way for a more concentrated and pertinent analysis.

The `clone_model` method is invoked to engender a compartmentalised replica of the universal model. This is realised by systematically iterating over each metabolite in the model and instantiating a new entity for every compartment wherein the metabolite is found, as elucidated in the `clone_metabolite` function code snippet below. Each cloned metabolite is bequeathed a distinct ID and compartment, guaranteeing its unique identification within each compartment of the model.

```
def clone_metabolite(compartments, metabolites):
    cloned_metabolites = []
    for metabolite in metabolites:
        for compartment in compartments:
            cloned_metabolite = metabolite.copy()
            cloned_metabolite.id = '__'.join(metabolite.id.split("__")[:-1]) +
                                                        '__' + compartment
            cloned_metabolite.compartment = compartment
            cloned_metabolites.append(cloned_metabolite)
    return cloned_metabolites
```

Developing the Custom Universal Model per se

For the next phase, the `build_custom_universal_model` method is central. The reader may refer to Figure 5 for a visual representation of this process. The method constructs the Custom Universal Model *per se*, it commences by identifying certain pathways to ignore while initialising an empty list for pathways to retain and creating a map of metabolite pathways.

The specifically chosen pathways to ignore include: 'Biosynthesis of secondary metabolites', 'Microbial metabolism in diverse environments', 'Biosynthesis of cofactors', 'Carbon metabolism', and 'Fatty acid metabolism'. These pathways are generally excluded due to their broad or non-specific nature. The exclusion of these pathways is a strategic decision to enhance the focus and specificity of the custom

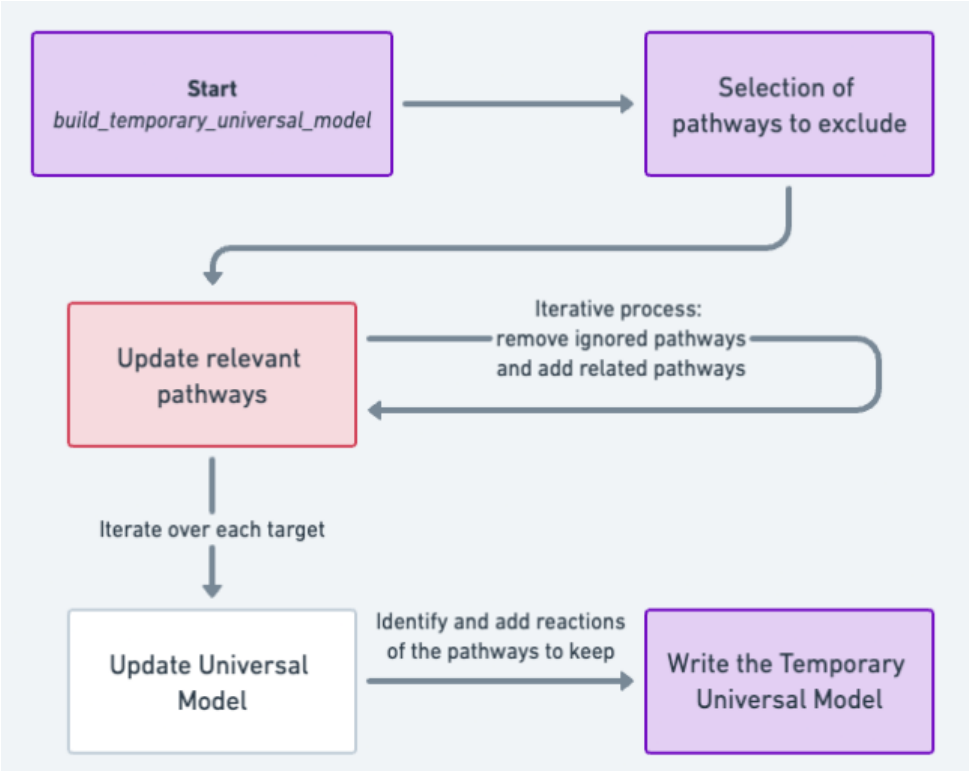


Figure 5: Diagram illustrating the steps for developing the Custom Universal Model.

universal model. For instance, the 'Biosynthesis of secondary metabolites' pathway encompasses a wide range of diverse metabolic pathways, many of which may not be relevant to the specific metabolic network under study. Ignoring such broad pathways allows the algorithm to concentrate on more specific and relevant pathways. This refined focus not only improves efficiency but also effectively reduces the search space, ensuring a more targeted and efficient approach. Similarly, pathways like 'Biosynthesis of cofactors', 'Carbon metabolism', and 'Fatty acid metabolism' are also excluded for their extensive and varied reactions, which may not be applicable to the specific metabolic network being reconstructed. This selective exclusion of pathways ensures that the custom universal model is constructed with reactions and metabolites that are most relevant to the specific metabolic network, contributing to a more efficient and focused gap-filling process. By excluding these generalised pathways, the algorithm is better positioned to hone in on the more specific and relevant pathways, ensuring a more efficient and targeted search for reactions that can effectively bridge the gaps in the metabolic network. The method then iterates over the targets' metabolites. For each metabolite, it checks its presence in the universal model's metabolite pathway map. If present, the pathways related to the metabolite are added to the pathways to keep, enhancing the specificity and relevance of the custom universal model to the metabolic network under study. After determining the pathways to keep, the method updates them by removing the ignored pathways and adding related pathways if they exist. It then updates the metabolite pathways map with related pathways,

ensuring that the map only contains pathways that are relevant and specific to the metabolic network under study. The method proceeds to identify the reactions to keep by iterating over the universal model's groups and reactions. It adds only those reactions that are part of the pathways to keep to the custom universal model. This step ensures that the custom universal model only contains reactions that are relevant to the specific metabolic network, contributing to a more efficient search space for the next step of this workflow.

Finally, the method writes the custom universal model as an SBML model in the specified folder path. This step finalises the construction of the custom universal model, ensuring that it is ready for use in the subsequent steps of the gap-filling process.

This procedure guarantees the creation of a universal model tailored to the specific metabolic network. By ensuring only relevant reactions and metabolites are included, it streamlines the gap-filling process. This refined model, referred to as the custom universal model, serves as the definitive file for the Meneco (1) algorithm. It acts as the pool containing all potential reactions for solving our gap-filling problem for a specific draft model.

3.2.4 Utilising the Meneco algorithm

Within the methodology of this research, the Meneco (1) algorithm is crucial for identifying potential reactions to address the gap-filling problem in metabolic networks. The `GapFiller` object, equipped with draft, seeds, and targets, facilitates the integration of Meneco. The developed methods within the `GapFiller` class ensure the seamless operation of the entire process using Meneco's algorithm as depicted in Figure 6. These processes are elaborated upon below. Important to mention again that the *GapFiller* class implementation can be accessed at this link.

The *run* method is responsible for discerning unproducible targets within the draft network using the `get_unproducible` method. The outcomes of this operation are both displayed and stored in the `unproducible` variable for future reference.

To enhance the comprehensiveness of the analysis, the universal model (note that here the Custom Universal Model is being used), termed as `repairnet` at the code, is read from its SBML file. This `repairnet` is then combined with the draft network, resulting in a combined network, named at Meneco (1) as `combinet`. This combined network is then subjected to the `get_unproducible` method to identify unreconstructable targets. The findings from this step are displayed and archived in the `never_producible` variable. In scenarios where certain targets are deemed 'unreconstructable', the `identify_additional_seeds` method is invoked. This method is competent at expanding the seeds list by identifying potential cofactors from pathways associated with unproducible metabolites. Following

the identification and addition of these seeds, the seeds are re-read from the SBML file, and the combined network is re-evaluated to discern any remaining unreconstructable targets.

The next phase involves distinguishing between reconstructable targets. This is achieved by subtracting the set of unreconstructable targets from the unproducible targets. The outcomes of this operation are displayed and stored in the `reconstructable_targets` variable. With the targets clearly determined, the `get_minimal_completion_size` method is invoked. This method's objective is to identify the smallest set of reactions that, when integrated into the draft model, would facilitate the production of the reconstructable targets. The results of this operation are displayed and stored in the `minimal_completion` variable.

To ensure a holistic solution, the `get_optimal_completions` method is used. This method explores multiple sets of reactions that can achieve the target, with the number of sets it should consider being determined by the `max_solutions` input value.

Upon the completion of the gap-filling algorithm, the total time taken for its execution is displayed. The identified reactions are then integrated into the model using the `add_reactions_to_model` method. If a particular solution doesn't lead to biomass production, the algorithm introduces demand reactions for all the identified dead-ends in the metabolic network. These demand reactions simulate the consumption or production of specific metabolites, potentially rectifying the inability to produce biomass. By dynamically adding and pruning these demand reactions, the model can more effectively identify gaps in the metabolic network and suggest appropriate reactions to fill these gaps, ensuring that the model remains biologically relevant and avoiding unnecessary complexity.

The final phase involves the calculation of the growth rate of the model using the `slim_optimize` method. If the optimisation yields a positive result, any redundant seeds and demands are pruned from the model. A report of the whole process is then generated as a JSON file.

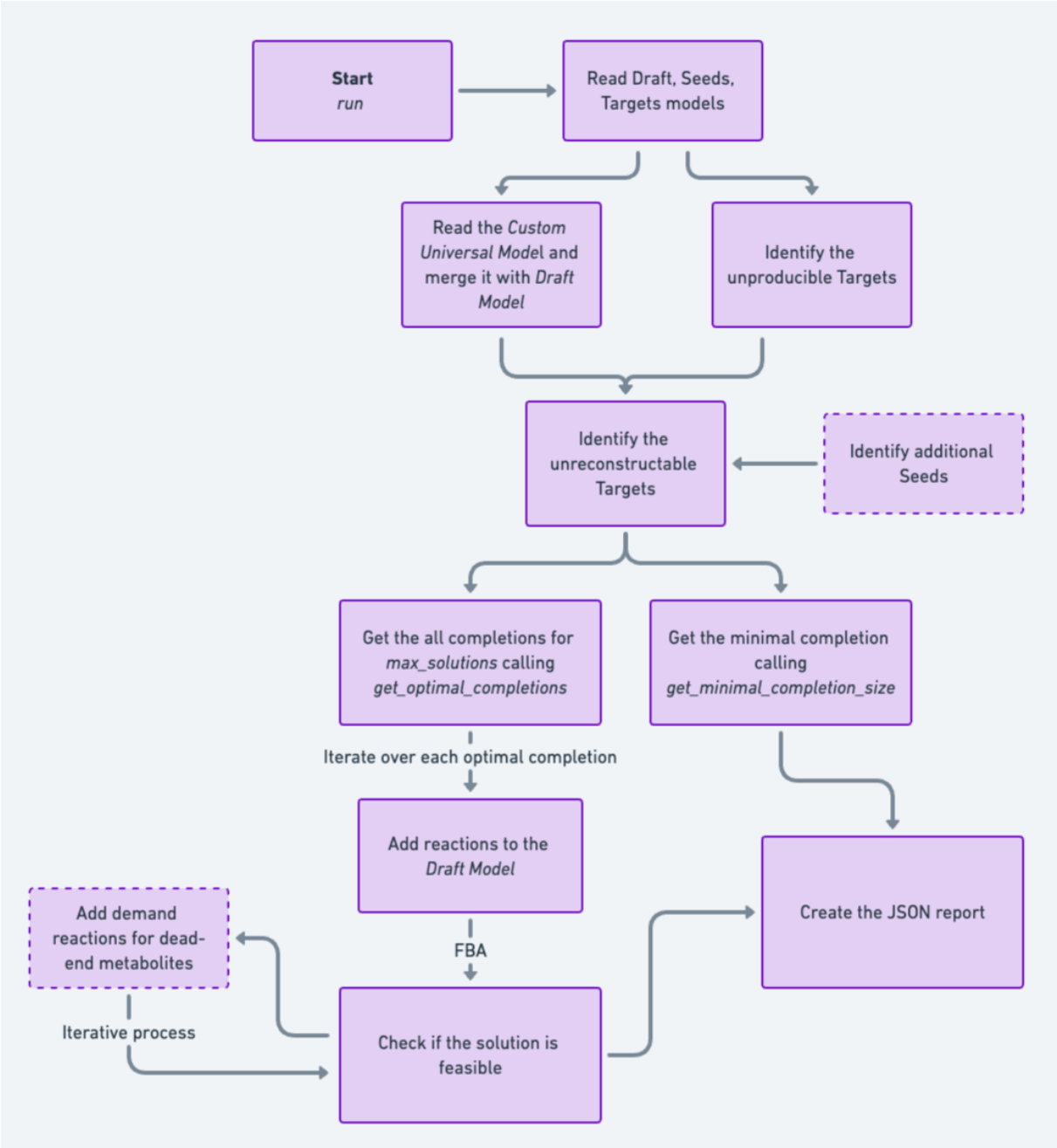


Figure 6: Diagram illustrating the integration of Meneco's algorithm with the developed methods to ensure feasible final solutions.

Report Structure

The structure of the reports generated after using the whole workflow (1) is presented below. These reports are formatted in JSON, facilitating easy parsing and subsequent analysis. Each report offers a comprehensive snapshot of the gap-filling process, detailing the files utilised, execution time, and the reactions and metabolites involved.

- **Title:** A simple heading, "Gap-filling report", denotes the content of the report.
- **Files Used:** Specifies the files employed during the process, which include the model, seeds, targets, and a custom universal model, each identified by their respective filenames.
- **Execution Time:** Captures the total duration, in seconds, taken for the algorithm's execution.
- **Artificial Removed Reactions:** Enumerates any reactions that were artificially removed, providing their specific identifiers.
- **Unproducible Targets:** Lists the targets that remained unproducible, accompanied by their identifiers.
- **Unreconstructable Targets:** Details the targets that could not be reconstructed.
- **Reconstructable Targets:** Enumerates the targets that were successfully reconstructed, along with their identifiers.
- **Minimal Completion:** Provides a list of reactions that were considered for the minimal completion of the network.
- **Additional Seeds:** Enumerates any supplementary seeds identified during the process, subsequent to their initial recognition using the `Model` class.
- **Essential Additional Seeds:** These are the seeds that were deemed crucial for the metabolic network's functionality. They are identified after the initial seeds have been added and are essential for ensuring that the model can produce its objective.
- **Additional Demands:** This section lists the demand reactions that were added to the model to address dead ends. Demand reactions are pseudo-reactions that allow a specific metabolite to be produced without being consumed elsewhere in the model. They are typically added to ensure that all metabolites in the model have a clear path and can be produced or consumed as required.

- **Summary:** Provides a succinct overview of the filled model. This section details the reactions and their associated metabolites, the factors, and the flux values.

- **Positive Solutions:** Enumerates the solutions that successfully produce biomass when evaluated using FBA. These solutions represent viable modifications to the metabolic model, ensuring that the primary objective of biomass production is achieved. Each solution in this category has been verified to enhance the model's capability to simulate the desired metabolic activity.

This structured report provides a solid record, facilitating the examination and assessment of the gap-filling procedure throughout the complete workflow, especially post-execution of the Meneco (1) algorithm.

3.2.5 Required and Generated Files and Their Significance

During the testing process, a series of files were generated, each holding significant value in understanding and analysing the performance of the developed methods. Below, we delineate the role and importance of each file:

- **Metabolic Model File:** This XML (128) file contains the structure of the metabolic model post the introduction of artificial gaps. It serves as the primary file where the modified metabolic pathways are stored, ready for analysis and testing.

- **Seeds File:** This XML file (128) houses the information pertaining to the seeds present in the modified metabolic model. Seeds are essentially the starting points in the metabolic pathways, and this file helps in understanding how the artificial gaps have influenced the initial points of these pathways.

- **Targets File:** Similar to the seeds file, this XML (128) file contains data on the targets in the metabolic model, which are the endpoints of the metabolic pathways. Analysing this file offers insights into how the artificial gaps have affected the final products in the metabolic pathways. This file is constructed after the gaps are introduced, gathering the targets related to the artificial gaps, and providing a perspective on the potential impacts and the areas requiring attention.

- **Universal Model File:** This file, formatted in XML (128), encapsulates the universal model derived from KEGG. It plays a key role by providing a blueprint of universal metabolic pathways. Serving as a foundational reference during both the testing and analysis phases, this file offers an exhaustive overview of potential metabolic pathways. Furthermore, it encompasses all reactions catalogued in the database, ensuring that users have access to the most up-to-date information.

- **Universal Model Compartmentalised File:** This XML (128) file houses a compartmentalised version of the universal model, dividing it into distinct compartments representing different cellular locations. This organisation aids in a more precise analysis during the gap-filling process. The file lays the groundwork for constructing the next described file.

- **Custom Universal Model File:** This file holds the custom universal model created during the workflow process. It originates from the universal model but contains only the reactions pertinent to the pathways of the organism under study and the identified targets, serving as a tailored reference point during the analysis for the gap-filling algorithm of Meneco (1) to have those reactions as reference. This is the file used as input for Meneco's algorithm, serving as a "repair net".

- **Report File:** This JSON file encapsulates a report generated after the gap-filling process. It contains vital data. This structured report stands as a testament to the method's efficacy, offering a detailed insight into the outcomes of the gap-filling process and aiding researchers in analysing the method's performance.

Each of these files plays a crucial role in the testing framework, offering a deep insight into different facets of the metabolic model and the performance of methods attached to the developed workflow. They serve as rich resources for researchers aiming to delve deep into the intricacies of the method's performance.

3.3 Utilisation of Docker for Software Containerisation

In the present research, Docker (129), a widely used open-source platform, was employed to facilitate the development and deployment of computational tools and services. Docker (129) utilises Operating System (OS)-level virtualisation to deliver software in packages termed as containers. These containers encapsulate the software within a comprehensive filesystem that encompasses everything requisite for its operation, including the code, runtime, system tools, and system libraries. This approach ensures that the software operates reliably and consistently across different computing environments by encapsulating not only the application but also its dependencies and configurations within a container (130).

The Dockerfile script, crafted for this project and accessible via the following link: Dockerfile, delineates the systematic procedure for configuring the Docker (129) container. The process initiates with the designation of the Ubuntu 22.04 image as the foundational layer. Subsequently, essential project files and software dependencies, inclusive of Python 3.10 and a variety of Python packages, are installed within

the container. A significant step in the setup is the installation of the BioISO tool through a Git clone followed by its setup using Python. Moreover, the CPLEX optimiser, a sophisticated solver for mathematical programming, is integrated to facilitate optimisation routines pivotal in metabolic network analysis (131).

Within the utilities folder, several essential files are maintained to support the functioning of the system:

- **cofactors.json:** A JSON file containing data on various cofactors involved in the metabolic pathways.
- **cplex_response.txt:** A text file storing responses for the CPLEX optimiser during the installation process.
- **cplex_studio2210.linux_x86_64.bin:** The binary file is necessary for the installation and functioning of the CPLEX optimiser in a Linux environment.
- **related_pathways_map.json:** A JSON file holding the mapping information of related pathways, facilitating the understanding and analysis of metabolic networks.
- **universal_model.xml:** An Extensible Markup Language (XML) file representing the universal model derived from KEGG (47), serving as a reference in metabolic network analyses.

The adoption of Docker (129) bestows several scientific advantages, enumerated as follows:

- **Reproducibility**: Docker (129) enhances the reproducibility of computational experiments by encapsulating the entire runtime environment, thereby ensuring uniform software behaviour across diverse computing environments (132).
- **Isolation**: The platform offers an isolated environment for the software, safeguarding it from potential conflicts with the system libraries of the host system and guaranteeing a pristine, controlled environment for software operation.
- **Portability**: Docker (129) containers, renowned for their ease of sharing and deployment across varied systems, foster collaboration and streamline the deployment process in disparate environments.
- **Version Control**: Docker (129) facilitates efficient tracking and management of updates and alterations through its version control feature, enhancing the manageability of the environment.

Within the Dockerfile, the Python path is augmented to incorporate the source directory, and the working directory in the container is designated as `/workdir`, thereby establishing a structured and

organised workspace. Moreover, the container is configured to expose port 80, enabling communication with the services operational within the container.

The utilisation of Docker (129) in this research not only streamlined the setup and deployment process but also fostered a reproducible and controlled computational environment, which is pivotal in scientific research to ensure the validity and reliability of the results.

3.4 Performance tests

To evaluate the performance of the workflow, which incorporated both Meneco's (1) fastest method (`get_minimal_completion_size`, code available here), BioISO and the developed methods, the `create_random_knockout_models` function from the `Model` class available at this link was responsible to produce artificial gaps. Leading to the creation of eight distinct test models, each characterised by its unique set of artificial gaps. The random knockout models ranged between three to six induced gaps, assuring no biomass production for each case. The diversity of these test models ensured a thorough evaluation, encompassing a wide range of potential scenarios. The curated model of *Streptococcus pneumoniae* R6, used for this study, was obtained from Dias et al., 2019. It can be accessed using the following DOI: 10.3389/fmicb.2019.01283. The growth medium and additional pertinent details for this model are available in the supplementary materials of the mentioned article.

In addition to the curated model, three draft models generated using *merlin* (2) were incorporated into the testing framework. These draft models, inherently characterised by their gaps, served as ideal candidates to challenge and validate the developed methods. Their inclusion ensured that the workflow was tested not only against a refined model but also against models that are more representative of initial draft reconstructions, which are often riddled with gaps and inconsistencies. Below is a brief description of the organisms used.

- *Lactococcus lactis*: This Gram-positive bacterium is of great value to the dairy industry. Recognised as a facultative anaerobic lactic acid bacterium, it thrives in environments devoid of oxygen, playing an integral role in the fermentation processes of dairy products. Beyond its industrial applications, *L. lactis* has garnered attention in biotechnological research, especially for its capability to produce bioactive peptides with health-enhancing attributes (133). Serving as a model organism, it offers insights into the metabolic processes of lactic acid bacteria. The accession number is: `GCA_023343905.1`

- *Synechocystis* sp.: This cyanobacterium is known for its photosynthetic capabilities. It is renowned

for its ability to fix atmospheric nitrogen and has been studied for its potential in biofuel production and carbon dioxide sequestration (134; 135). The metabolic pathways of *Synechosystis* are of interest due to their unique photosynthetic capabilities and their potential role in sustainable energy solutions (136). The accession number is: `GCA_000009425.1`

- *Chlorella vulgaris*: A freshwater green microalgae known for its high protein content. It has also been studied for its potential in biofuel production, wastewater treatment, and as a source of nutritional supplements (137; 138). The metabolic pathways of *C. vulgaris* are of interest due to their ability to produce lipids, which can be converted into biodiesel (134). The accession number for this organism is: `GCF_000009725.1`

The aforementioned models were utilised to execute the developed workflow, as detailed in the preceding sections of this chapter. All of these models were subsequently contrasted with the typical procedures that rely solely on Meneco (1). In this context, the targets used were those associated with the biomass reaction. Meanwhile, the input of seeds was those identified within the developed workflow. This approach was taken because, without these seeds, the targets would not be reconstructable by the algorithm.

The hardware specifications used to run both of the workflows (the developed one under this study and Meneco's workflow) are as follows:

- Model: SUPERMICRO SYS-6049GP-TRT
- Processor: CPU INTEL XEON SILVER 4216 16C/32T 2.1GHZ 22MB LGA3647 (x2)
- Operating System: CentOS 8 Stable
- RAM: 251 GB

3.4.1 Case studies

As previously mentioned, the *Streptococcus pneumoniae* R6 model underwent artificial gap inductions, resulting in the creation of eight distinct models. These models form a part of the case studies. In addition to these, three other draft models were developed and are also included in the case studies. As highlighted in the State-of-the-art chapter, several tools exist for the automatic reconstruction of a GEM. However, these draft models often require further refinement and curation. In this work, *merlin* was the primary tool employed for generating the models (2). *merlin* offers a suite of features, primarily as *plugins*, to efficiently and swiftly produce these models. Detailed descriptions of these features have been provided

earlier in this document. For a comprehensive understanding of each step, it is recommended to refer to the GEM reconstruction-related sections in the State-of-the-art chapter. Below, the procedures undertaken to generate the draft models for this study are succinctly outlined.

Genome annotation

The reconstruction process starts with the functional annotation of the genome, wherein the functions of genes are predicted based on their sequences. For this research, DIAMOND, integrated within *merlin* (2), was employed to identify homologous sequences. The homology search was performed against Swiss-Prot using an e-value threshold of $1e^{-30}$.

Draft assembly

Following genome annotation, a draft metabolic network is constructed, serving as an initial version of the GEM. During this phase, *merlin* assembles the metabolic network by associating the EC numbers assigned before with their corresponding reactions. Reaction details, such as stoichiometry, and cofactors are retrieved from KEGG (47). The reversibility was corrected using the *merlin*'s "Correct reversibility" tool with MODELSEED (139) as a template, and the transport reactions were predicted using TranSyT (50). Additional simple diffusion transport reactions for water, oxygen, carbon dioxide, ammonia (and photons in the case of *Synechocystis* sp. and *C. vulgaris*) were added to the models, as TranSyT does not automatically include them. The subcellular location for *L. lactis* and *Synechocystis* sp., was predicted with PSORTb 3.0 (140), while for *C. vulgaris* LocTree 3 (53) was used. A biomass equation was retrieved from available GEMs for each organism, as well as the respective growth media. This assembly process ensures that the draft model is a comprehensive representation of the organism's metabolic capabilities, albeit with potential gaps or missing reactions. The biomass composition was retrieved from the publications (141; 142; 143). A table representing the biomass composition for each organism is provided in Table S1. The parameter "max_solutions" was fixed at 50 to limit the maximum number of completions provided by Meneco.

Gap-filling with Draft Models

Whilst the subsequent step in many workflows would be network curation, at this juncture, the model is replete with gaps. These gaps highlight areas of the metabolic network that are not yet fully understood or represented. The draft models are now primed for the gap-filling procedure, representing the importance of generating these preliminary models under this study; these will be utilised in the subsequent stages of the workflow, where the gap-filling designed and implemented methods will be applied. For all the models,

the `objective_function_id` set in the JSON parameters file was `e_Biomass__cytop`.

The growth media for each organism, as detailed in Table S2, represents the specific metabolites essential for their growth and metabolic activities. The value of 1000 in the table indicates the maximum uptake rate of these compounds. This high value ensures that the uptake of a compound isn't artificially limited in preliminary models, especially when exact rates aren't known.

- *Lactococcus lactis*: The draft model's growth medium for this bacterium is notably enriched, featuring a variety of amino acids such as L-Glutamate, Methionine, L-Arginine, and L-Valine. Additionally, it incorporates vital metabolites like Adenine, Uracil, Cytosine, Guanine, Glucose, Water, and Diphosphate, as detailed in Table S2. The deliberate exclusion of oxygen from its growth medium aligns with its facultative anaerobic nature. The biomass equation, specifics of the growth medium, and model parameters were derived from a well-established metabolic model (141). The model predominantly focuses on a singular compartment, the `cytop` (cytoplasm), as reflected in the parameters.

- *Synechosystis* sp.: The growth medium for this cyanobacterium's draft model is primarily composed of inorganic compounds, including Water, Proton, Sulfuric acid, Phosphate, Nitrate, Ferrous ion, Carbon dioxide, Oxygen, and Ammonia, among others. This minimalist inorganic medium underscores its capability to harness essential nutrients from rudimentary sources, complemented by its photosynthetic prowess. The biomass equation, growth media, and model parameters were informed by a pre-existing metabolic model (142). The biomass composition, which is detailed in Table S1, reflects its unique metabolic capabilities. The model exclusively replicates the `cytop` compartment and incorporates sinks for ACP (`C00173__cytop`) and tetrahydrofolate (`C00101__cytop`).

- *Chlorella vulgaris*: The growth medium for this green microalga's draft model encompasses Water, Proton, Sulfuric acid, Phosphate, Ferrous ion, Carbon dioxide, Photon, Chloride, and other metabolites, as elucidated in Table S2. This assortment of inorganic compounds mirrors its photosynthetic attributes and its aquatic habitat. The biomass equation, growth media, and model parameters were inspired by a previously published metabolic model (143). The workflow's design involved cloning multiple compartments: `cytop` (cytoplasm), `pero` (peroxisome), `chlo` (chloroplast), `mito` (mitochondria), `er` (endoplasmic reticulum), and `golg` (Golgi apparatus). To account for certain long- and very-long-chain fatty acids in the biomass whose synthesis pathways are not fully represented in KEGG, sinks were added. Specifically, the sinks added were: `C06427__cytop`,

C06425__cytop, C01595__cytop, C08281__cytop, C16525__cytop, C16535__cytop, C16537__cytop, C21944__cytop, and C00219__cytop.

Chapter 4

Results and Discussion

4.1 Generated draft models overview

The models selected for this study represent a diverse set of organisms, each with its unique metabolic distinctive characteristics, allowing to evaluate the workflow's performance in different contexts. Table 3 shows a summary of the main statistics of the models used here.

To evaluate if the workflow is able to provide accurate solutions, the *Streptococcus pneumoniae* R6 model is a curated model retrieved from literature to which artificial gaps were introduced to simulate the challenges typically associated with draft models. Then, the workflow was applied to real-case scenarios to evaluate the efficiency of the developed workflow in comparison to a well-established approach that relies solely on Meneco's (1) workflow. Thus, three draft models were generated using *merlin* (2), namely for *Lactococcus lactis*, *Synechocystis* sp., and *Chlorella vulgaris*. The forthcoming sections will delve deeper into the specifics of our gap-filling process, highlighting the effectiveness of this workflow.

Table 3: Number of reactions, metabolites, genes, compartments and pathways of the models used as subjects for the developed workflow. The *S. pneumoniae* R6 model is related to a high-quality published GEM for this species, while the remaining ones refer to draft models developed in this work.

Organism model	Reactions	Metabolites	Genes	Compartments	Pathways
Streptococcus pneumoniae R6	596	489	372	1	40
Lactococcus lactis	1285	1158	514	1	120
Synechocystis sp.	1455	1456	621	4	134
Chlorella vulgaris	6482	5984	1674	9	144

4.2 Performance test results: Developed Workflow

This section evaluates the performance of the workflow devised for predicting the minimal set required for gap-filling across a range of models, as detailed in the preceding sections of this document. BioISO was employed to identify the targets. Furthermore, Meneco (1) was harnessed to streamline the gap-filling process, providing the algorithm to ascertain the minimal set. Particularly, the union of BioISO (11) and Meneco leads to the term *BioMeneco*, as delineated by Cruz *et al.* (2021) (11), manifesting the collaborative potency of these tools within our workflow.

Firstly, the seeds and targets identified by the workflow were evaluated. Subsequently, the custom universal model is assessed, followed by an examination of the final solution derived from Meneco's results, and the respective execution time. The subsequent subsections provide detailed results for each model used to test the workflow, starting with a more independent analysis for the models with induced gaps for the *Streptococcus pneumoniae* R6 model, followed by an evaluation of each draft model generated when submitted to the whole workflow.

4.2.1 Induced Artificial Gaps

To allow evaluating the workflow's performance and accuracy, random artificial gaps were introduced into the *Streptococcus pneumoniae R6* model, resulting in eight distinct models: `model_1` through `model_8`. Table S3 offers an in-depth overview of these models, detailing the specific reactions removed to induce the gaps.

This section provides an analysis of the identified seed and target metabolites, alongside other pertinent results from the workflow developed for the eight artificially-gapped models of *Streptococcus pneumoniae* R6. The identification of seeds and targets was successful in all models. A natural pattern is the localisation of the majority of seed metabolites externally to the cell, while target metabolites were all found within the cytoplasm for the *Streptococcus pneumoniae* R6. This distribution is coherent with the explanation of the seeds and targets term meaning as previously discussed in this document.

In `model_1`, six target metabolites were identified within the cytoplasmic compartment, namely monoglucosyldiglyceride (C04046), fatty acid (C00162), glycerol (C00116), phosphatidylglycerol (C00344), cardiolipin (C05980) and diglucosyldiacylglycerol (C06040). After introducing artificial gaps these metabolites cannot be produced. A detailed list of these metabolites is provided in Table S4. Similarly, for `model_2`, seven target metabolites were identified (Table S5). The models `model_3` to `model_8` display a variety of target metabolites within the *cytoplasmic* compartment. The subtle modifications in each

model lead to diverse metabolic outcomes, as seen in Tables S6 to S11. Glycerol (*C00116*) was identified as a target in all models. The reaction producing this compound in the original model (reaction ID *R07390_C3_cytop*) is responsible for the production of cardiolipin from phosphatidylglycerol, releasing glycerol. Since cardiolipin is a biomass component, and there are no reactions consuming this compound other than the biomass reaction, the reaction *R07390_C3_cytop* can only have flux when all biomass components are produced.

Table 4 shows a comprehensive summary of the results related to the final generated report for the eight gapped models, offering an in-depth overview of the gap-filling properties and computational characteristics of each model applied to the developed workflow. The table delineates the missing reactions in the metabolic networks, those that were artificially removed, as well as the number of target metabolites identified for each model. Reconstructable targets provide insights into potential refinements of the model, indicating target metabolites that become producible upon the introduction of certain reactions into the model. The term "Minimal Completion Size" denotes the minimum number of reactions that the algorithm identified as essential to ensure the producibility of the reconstructable target metabolites, essentially representing the minimal solution found by the algorithm for the model to achieve a fully functional metabolic network. The execution times for all these models were reasonably low. The growth rates across most models were consistent and aligned with the original model's growth rate of $1.21h^{-1}$, except for `model_4`, which exhibited a reduced growth rate of $0.39h^{-1}$. Regarding the additional seeds, as mentioned in the Methods chapter, when a target is identified by the algorithm as unreconstructable, a method developed will attempt to find new seeds in order to render this target as "reconstructable". All models were gap-filled without needing for additional seeds or demands, as the number of reactions removed is relatively low. `Model_5` is the only exception, requiring *NADP+* as a seed. This can be explained by the absence of one artificially removed reaction (*R01706*) in the custom universal model, making the targets unreconstructable. As explained previously, the workflow tries to add new seeds associated with cofactors to allow the reconstructibility of the identified targets. This result indicates that filtering the universal model with the pathways associated to the targets may lead to less accurate solutions in some extreme cases, despite the reduction in the running time (results in further section).

Table 4: Summary of results from the developed workflow when applied to artificially induced gap models of eight *Streptococcus pneumoniae* R6 reaction knockouts.

Model	Execution time (s)	Growth rate (h^{-1})	Rec[a] : Unrec[b] Targets	Minimal Completion Size	Essential Seeds and Demands
`model_1`	9.25	1.21	6:0	2	0
`model_2`	10.02	1.21	7:0	5	0
`model_3`	9.27	1.21	7:0	3	0
`model_4`	8.64	0.39	5:0	2	0
`model_5`	8.57	1.21	2:0	1	1
`model_6`	8.87	1.21	2:0	2	0
`model_7`	9.48	1.21	2:0	2	0
`model_8`	9.20	1.21	8:0	3	0

[a] Reconstructable [b] Unreconstructable

Analysis of Minimal Set vs Removed Reactions

An important aspect of the workflow's accuracy is the comparison between the reactions that were artificially removed and the reactions that the workflow suggests adding back to restore the model's functionality. This comparison provides insights into the accuracy and precision of the workflow in identifying and rectifying the artificially induced gaps.

As mentioned, for each model, a set of reactions was removed to introduce gaps, and the workflow subsequently identified a set of reactions to be added (the minimal set) to restore the model's functionality. The reactions that match between the removed set and the minimal set indicate the workflow's ability to correctly identify and suggest the reintroduction of reactions that were artificially removed. Table 5 presents the results of this comparison. For instance, in `model_1`, the workflow correctly identified `R04428` for addition, matching it with the reactions that were initially removed. Similarly, for `model_2`, the workflow accurately identified `R02029`, `R04968`, and `R01123`. Particularly, in the case of `model_2`, within the "all completions" calculated, the reaction `R04724` was also identified as a match. Regarding the remaining models, `model_3`, `model_4`, `model_5`,`model_6`, `model_7`, and `model_8`, the workflow managed to reproduce the same patterns in successfully identifying part of the removed reactions. Specifically, for `model_3`, the workflow's suggestions of `R06447` and `R04960` aligned with the reactions that were removed. For `model_4`, the reaction `R09381` was both removed and subsequently identified by the

workflow to be added back. `Model_6` had the accurate identification of R04377. In `model_7`, R01658 was correctly identified by the workflow. For `model_8`, both R09380 and R00416 were correctly identified by the workflow.

However, it's noteworthy that the number of reactions in the minimal set is often fewer than the number of removed reactions. This discrepancy is expected, as not every removed reaction necessarily leads to dead-ends in the metabolic network.

Additionally, the KEGG's database characteristics lead to potential mismatches between the reactions removed and the meneco result. For example, the reaction *R04724* was artificially removed, and the reaction *R04725* was suggested to be added. These reactions belong to the "Fatty acid biosynthesis" pathway, and both convert trans-Dodec-2-enoyl-[acp] to Dodecanoyl-[acyl-carrier protein]. The difference between the reactions relies on the cofactor utilisation: while *R04724* uses NADH/NAD+, *R04725* uses NADPH/NADP+. Moreover, KEGG's database occasionally combines multi-step reactions into a single reaction or represents them as a set of individual reactions, each delineating a distinct step in the overall process. Although this phenomenon was not observed in this specific case study, it is important to be aware of this possibility. In these cases, the optimization algorithm will always choose the single reaction, as it minimizes the number of added reactions.

Table 5: Comparison between artificially removed reactions and Workflow-Recommended additions for the knockout models of *Streptococcus pneumoniae* R6.

Model	Removed Reactions	Minimal set
`model_1`	R00239, R04559, R02294, R03509, **R04428***	**R04428***, R11104
`model_2`	**R02029***, R04724, R02018, **R04968***, **R01123***, R02323	**R02029***, **R01123***, R11104, R04725, **R04968***
`model_3`	R04426, R01773, R00803, **R06447***, R02019, **R04960***	R07636, **R06447***, **R04960***
`model_4`	**R09381***, R01715, R01061, R00260, R05068, R00480	R00160, **R09381***
`model_5`	R00104, R01706, R00239	R11104
`model_6`	R01397, **R04377***, R10147	R00160, **R04377***
`model_7`	R06863, R05069, R02295, R02569, **R01658***	R11104, **R01658***
`model_8`	**R09380***, **R00416***, R02295, R00573, R01220	R07636, **R09380***, **R00416***

* Matching reaction

4.2.2 Draft models: Seeds and Targets identification

This section is related to the results of the step of identification of seeds and targets for the draft models generated with *merlin* (2). Table 6 summarises the seeds and targets identified within the draft networks of *L. lactis*, *Synechocystis* sp., and *C. vulgaris*.

The simplest case study with draft models refers to *L. lactis* model. The draft model of this species has only the cytoplasm and extracellular environment, and the growth media contains a carbon source and amino acids. Table S12 in the appendices offers a detailed overview of the identified seeds and targets within the *L. lactis* metabolic model. This table classifies metabolites based on their respective compartments and provides their associated metabolite IDs. These seeds span across cytoplasm and extracellular space. Furthermore, specific target metabolites were identified within the cytoplasmic compartment, each associated with various metabolic pathways. In total, were identified 47 seeds and 28 targets.

More detailed information is also available for *Synechocystis* sp. in Table S14. In total, 35 initial Seeds were identified in the *Synechocystis* sp. metabolic model. Along the workflow, it was possible to identify extra additional seeds. Regarding the targets, 42 dead-end metabolites were found within the cytoplasm.

The draft model of *C. vulgaris* is the most complex case study. As in *Synechocystis* sp, the media for this algae only contains inorganic compounds. Since it is an eukaryotic organism, it presents several subcellular compartments, including cytoplasm, mitochondria, chloroplast, endoplasmic reticulum, and peroxisome, in addition to the extracellular space. Table S16 in the appendices provides a comprehensive overview of the identified seeds and targets within the *C. vulgaris* metabolic model. This table classifies metabolites based on their respective compartments and provides the number of identified metabolites for each classification. Although most targets are located in the cytoplasm, the pigments (e.g., chlorophyll a, β-carotene) were identified in the chloroplast, since they were introduced in the biomass reaction with this compartment. In total, 43 seed metabolites were identified across multiple compartments and 47 target metabolites within the cytoplasm, chloroplast and endoplasmatic reticulum compartments in the *C. vulgaris* model.

4.2.3 Draft Models: Custom Universal Model

This section delineates the results garnered from the construction of the custom universal model, as explained in the Methods chapter. The universal model is a comprehensive representation encompassing

Table 6: Summary of Seeds and Targets identified for the draft models of *L. lactis*, *Synechocystis* sp. and *C. vulgaris*.

Model	**Total Seeds (Additional Seeds)**	**Compartments**	**Total Targets**	**Compartments**
Lactococcus lactis	47 (10)	cytop[a], extr[b]	28	cytop
Synechocystis sp.	54 (19)	cytop, extr	42	cytop
Chlorella vulgaris	221 (144)	cytop, extr, mito[c], chlo[d], er[e], pero[f]	48	cytop, chlo, er

[a] Cytoplasm[a] [b] Extracellular [c] Mitochondrion [d] Chloroplast [e] Endoplasmic Reticulum [f] Peroxisome

almost all reactions, metabolites, and pathways within the KEGG's database, encapsulating a total of 163 metabolic pathways. Shortly, a clone of the universal model is created, replicating the metabolites and reactions to the compartments of the specific metabolic network under study. Then, demand reactions are introduced to all KEGG metabolites that are only associated with one reaction. Transport reactions are added to all targets between all internal compartments and the cytoplasm. Subsequently, a custom universal model is built, filtering the pathways in the network associated with the previously identified targets for the draft model. The core statistics pertaining to this process are succinctly presented in Table 7.

This compartmentalised model for *L. lactis* has a reaction count of 13,796, which is higher than the Universal Model. This increase is attributed to the demand reactions that were added, as well as to the transport reactions that were included in the model. The custom universal model, tailored specifically for each draft model, comprises 11,730 reactions and 7,610 metabolites. This model narrows down the focus to 100 metabolic pathways that are related to *L. lactis*'s metabolism and the previously identified targets.

The custom universal model for *Synechocystis* sp., as summarised in Table 7, includes 13,3796 reactions and 13,311 metabolites distributed in one compartment. Although the draft model contains other compartments (extracellular, periplasmic space, and thylakoid lumen), only the cytoplasm was considered for the gap-filling, as these three compartments are associated with a low number of metabolic reactions. The number of reactions in the custom and compartmentalised model is similar. Since the number of targets in this species is high and includes biosynthesis of different biomass components (e.g., nucleotides, amino acids, lipids), most KEGG pathways were kept in the model, leading to a high number of reactions and metabolites.

The construction of the custom universal model for *C. vulgaris* involved filtering and honing the reactions and pathways from the Universal Model to focus on those imperative to *C. vulgaris*. As illustrated

in Table 7, the custom universal model encapsulates a total of 70,563 reactions spanning 106 metabolic pathways within 6 compartments, marking a significant reduction from the Universal Model. This construction and refinement process may validated through the decrement in the number of reactions, whilst still retaining a comprehensive representation of the target metabolic processes. In the custom universal model for *C. Vulgaris*, 225 transport reactions were added to address the gaps in the model. Given the complexities of metabolic networks related to transport reactions, the solution used in this workflow shows space for improvement. One possible solution is to use a database with transport reactions that can be added to the universal model. Although KEGG presents transport reactions, this database has its limitations in representing them, as only the most common reactions are present, and the retrieval from the KEGG's remote accession is different from the metabolic reactions. Another potential solution to further enhance the model's accuracy is to augment the data with information from other databases that present a wide set of transport reactions, such as ModelSEED or TranSyT. Incorporating data from these databases can provide a more robust view of the transport mechanisms, ensuring that the custom universal models are not only accurate but also exhaustive in their representation of metabolic processes.

The process of curating custom universal models from the overarching Universal Model is a meticulous endeavour aimed at honing the focus on pathways that are pertinent to the metabolic networks of the specific organisms under study. This tailored approach facilitates a more nuanced and organism-specific analysis, which is important for accurate gap-filling and subsequent metabolic reconstructions.

The reduction in the number of pathways in the custom universal models, as compared to the 163 pathways in the Universal Model, is a deliberate stratagem to shed the extraneous pathways that hold little to no relevance to the organisms in question. This reduction is emblematic of a more focused and streamlined model that accentuates the pathways integral to the metabolic networks of *L. lactis*, *Synechocystis* sp., and *C. vulgaris*.

Overall, the tailored nature of each custom universal model is exemplified by its alignment with the metabolic fragment of the network pertinent to the respective organisms. The Custom Universal Models for *L. lactis*, *Synechocystis* sp., and *C. vulgaris* encapsulated with success the relevant pathways associated with each draft model. Each of these models constitutes a focused subset of the extensive pathway repository present in the Universal Model, thereby showcasing a tailored approach to the metabolic specificities of each organism.

Table 7: Summary of the results on the development of the compartmentalised model and custom universal model for *Lactococcus lactis*, *Synechocystis* sp. and *Chlorella vulgaris*

Organism	Model	Reactions	TR[a]	Metabolites	Compartments	Pathways
-	Universal Model	10530	-	8677	-	163
Lactococcus lactis	Compartmentalised Model	13796	-	8665	1	163
	Custom Universal Model	11730	0	7610	1	100
Synechocystis sp.	Compartmentalised Model	13797	-	8667	1	163
	Custom Universal Model	13311	0	8470	1	142
Chlorella vulgaris	Compartmentalised Model	82728	-	51996	6	163
	Custom Universal Model	70563	225	45624	6	106

[a] Transport Reactions

4.2.4 Evaluation of Workflow's performance

The comparison of draft models for gap-filling under the developed workflow, presented in Table 8, shows the effectiveness of this process in the three case studies. The table summarises crucial metrics, including the number of reconstructable Targets, unreconstructable Targets, the number of inimal completion reactions, and the corresponding execution time. These metrics are all available in the JSON-generated report after the final workflow process.

For *L. lactis*, a total of 28 reconstructable Targets were identified, signifying the number of essential metabolic processes that could be successfully restored. To achieve a Minimal Completion with 24 reactions, the gap-filling algorithm necessitated an execution time of nearly 11 seconds. This demonstrates the efficiency of Meneco's algorithm when provided with well-established seeds, targets, and demand reactions offered by the workflow. Regarding *Synechocystis* sp., exhibited a notably higher number of reconstructable Targets, with all 42 targets successfully addressed. This suggests a more comprehensive biomass composition for this organism, resulting in a Minimal Completion of 63 reactions. However, this process took an amounting of approximately 69 minutes. This is related to the complexity and intricacy of the metabolic pathways in *Synechocystis* sp. and to the growth media selected for each organism. Unlike *L. lactis*, which has a medium enriched with glucose and amino acids, *Synechocystis* sp. relies predom-

inantly on inorganic compounds, such as orthophosphate, nitrate, and sulfate. Thus, the cyanobacteria require a more elaborate metabolic network to synthesise essential biomass components from inorganic precursors. The reliance on inorganic materials means that *Synechocystis* sp. has to perform more complex biochemical transformations, which in turn requires a larger number of reactions and pathways to be considered during the gap-filling process. This intricacy is reflected in the extended execution time of Meneco's algorithm for *Synechocystis* sp. compared to *L. lactis* for both workflows in comparison at Table 8 Similarly, *C. vulgaris* displayed a substantial number of reconstructable Targets, with 48 targets successfully reconstructed. This relatively efficient gap-filling process led to a Minimal Completion of 47 reactions. The execution time for this model was around 47 min, reflecting more complexity related to this metabolic network. For *Synechocystis* sp., the gap-filling process was more challenging. As this draft model had fewer seeds identified compared to *C. vulgaris*, the *Synechocystis* seemed to be intrinsically more "gapped", which led to a higher number of reactions at the minimal completion. Consequently, the execution time for *Synechocystis* was significantly longer than that for *C. vulgaris*, taking almost 70 minutes for the developed workflow.

It's essential to note that the growth rate values obtained, might seem unrealistic. This is because the exchange fluxes weren't limited, and this was not considered for sink reactions as well. However, the primary focus here is to ensure that the growth rate value is greater than zero, indicating a viable network. Furthermore, the developed workflow showed a growth rate greater than zero, while Meneco's approach did not. This discrepancy arises because the developed workflow incorporates demand reactions to address the false positives issue associated with Meneco, as discussed in the State-of-the-art.

Table 8: Comparison between the developed workflow and the typical Meneco approach for Gap-Filling in three utilised models, assessing Reconstructable Targets, Unreconstructable Targets, Minimal Completion Reactions, Growth rate (h^{-1}) and Execution Time (mm:ss:ms)

Workflow	**Model**	**Rec**[a] **: Unrec**[b] **Targets**	**Minimal Completion Reactions**	**Growth rate (h^{-1})**	**Execution Time (mm:ss:ms)**
Developed Workflow	*Lactococcus lactis*	28:0	24	37.72	00:10:946
	Synechocystis sp.	42:0	63	130.94	69:09:400
	Chlorella vulgaris	48:0	47	204.76	46:32:294
Meneco's Workflow	*Lactococcus lactis*	12:0	24	0.0	00:07:25
	Synechocystis sp.	6:0	54	0.0	335:30:159
	Chlorella vulgaris	7:0	42	0.0	60:03:481

[a] Reconstructable [b] Unreconstructable

In evaluating the gap-filling results from the developed workflow, detailed insights can be derived from the appendices containing targeted and identified reactions for each model organism. This in-depth analysis is essential to understand the specific metabolic pathways that have been impacted by the gap-filling process.

For *Lactococcus lactis*, the appendices Tables S12 and S13 shed light on the reconstructable targets, the additional seeds introduced, and the minimal set of reactions identified to fill the existing gaps in the metabolic network.

On the other hand, *Synechocystis* sp. exhibits a robust metabolic model with all 42 targets successfully addressed. Thus, a closer examination of Tables S14 and S15 reveals the specific reactions that were identified to achieve this completeness in the metabolic model. The tables also display the additional seeds which were crucial in realising the gap-filling objectives.

The analysis of *C. vulgaris* presented in Tables S16 and S17 offers an in-depth perspective on the reconstructable targets, seeds and additional seeds, and demand reactions. A significant number of additional seeds were necessitated for various compartments of the organism. The data also elucidates how the gap-filling algorithm navigated through multiple compartments, such as the cytoplasm, chloroplast, and endoplasmic reticulum, to ensure minimal completion. This suggests the presence of a more complex metabolic network. Detailed insights about the suggested minimal completion can be found in Table S18.

Limitations of the developed workflow

Lastly, it's imperative to acknowledge the limitations of this developed workflow. The first limitation is associated with the base universal model reconstruction. As mentioned before, the KEGG database, while extensive, presents a set of limitations for the purpose of this work. These include the presence of generic metabolites and reactions, incomplete reactions, missing or partial biosynthetic pathways, and ambiguities in reaction reversibility. The presence of generic compounds/reactions is not straightforward to solve, as removing those elements would make the optimisation problem unsolvable. Incomplete reactions were manually removed, solving this problem. However, a method to perform this task automatically must be developed to prevent this issue in future KEGG updates. Although KEGG is a comprehensive and complete biological database, it has deficiencies. Specifically, it presents incomplete pathways, particularly those related to long-chain fatty acids and lipids in general. Furthermore, the database lacks a robust structural definition for lipids. This shortcoming underscores the need for continuous refinement and expansion of the database to ensure its relevance and applicability in diverse research contexts.

Finally, the reaction's reversibility was corrected using the ModelSEED database. While ModelSEED is

a robust and comprehensive resource, it is not immune to potential limitations. There might be occasions when the database is not up-to-date, which can directly influence the outcomes of the workflow. For instance, certain reactions might be inaccurately represented, especially in terms of their reversibility. Such discrepancies can introduce challenges and uncertainties in the final results. Correcting the reversibility of reactions with another database, such as MetaCyc, could potentially solve this issue. Furthermore, while Meneco is a powerful tool for metabolic network completion, it has its limitations, as discussed in the State-of-the-art chapter. Some of these limitations have been addressed in the developed workflow through the development of specific methods. For example, some strategies were implemented to identify additional seeds and incorporate demand reactions, thereby enhancing the capabilities of Meneco and ensuring a more comprehensive and accurate gap-filling process. It is, therefore, crucial for users to be aware of the ongoing updates and discussions related to the ModelSEED database, as highlighted in their github repository, and to remain cognisant of the inherent challenges and solutions associated with tools like Meneco.

Chapter 5

Conclusions and Future Work

The research journey undertaken in this thesis was motivated by the pressing need for a more streamlined, efficient, and comprehensive approach to metabolic network reconstruction. The developed workflow, meticulously detailed in the preceding chapters, stands as a testament to the potential of harmoniously integrating computational methodologies with biological databases, such as KEGG, and tools like BioISO and Meneco.

When the results of the developed workflow are juxtaposed against the traditional Meneco approach, the advancements and improvements become evident. The preliminary results suggest that the workflow might possess the capability to handle a diverse array of organisms, ranging from the simpler metabolic networks of *L. lactis* to more complex ones. This potential versatility of the workflow warrants further exploration and validation. Besides, the observed efficiency in terms of time and computational resources is promising, but it would be essential to validate these findings with broader datasets and scenarios.

However, no scientific endeavour is devoid of challenges. The limitations encountered, containing the intricacies of the KEGG database and the inherent challenges of metabolic network reconstruction, were not just obstacles but also learning opportunities. They underscored the importance of adaptability and the need for continuous refinement in the realm of bioinformatics.

5.1 Future Directions

The developed workflow, while with significance, may offer several promising avenues to exploration:

1. **Experimental Validation:** The computational reconstructions, while robust, need to be anchored in biological reality. Laboratory-based experiments can serve as the definitive benchmark, validating the predictions and models generated by the workflow. Such validations would not only enhance the trustworthiness of the workflow but also provide invaluable feedback for further refinement.

2. **Database synergy:** Relying on a singular database, while effective, has its limitations. Integrating multiple databases, such as MetaCyc, BiGG (144), could offer a more panoramic view of metabolic pathways. Such integrations could mitigate the limitations of individual databases and provide a richer fountain of information.

3. **Algorithmic enhancements:** The core of the workflow, the Meneco algorithm, was developed for optimisation. Delving deeper into its intricacies, exploring alternative algorithms, or even enhancing its current capabilities could lead to even faster execution times and heightened accuracy.

4. **User-Centric development:** To democratise access to the workflow, a user-friendly interface, seamlessly integrated with *merlin*, would be invaluable. Such an interface would simplify data input, facilitate real-time monitoring, and offer intuitive visualisation tools, making the workflow accessible to a broader audience.

5. **Broadening the horizon:** The true test of the workflow's adaptability would be its application to a diverse array of organisms. Utilising the workflow beyond the organisms explored in this thesis and testing the workflow on both simpler and more complex organisms could offer deeper insights into its robustness and versatility.

5.2 Final Thoughts

The developed workflow represents decent progress in the domain of metabolic network reconstruction, especially when compared with the conventional approach of Meneco. As the field of Bioinformatics and Systems biology advance, methodologies and tools such as the one presented in this work become imperative for accurately aligning computational predictions with experimental biological data. While the challenges in this field are multifaceted, they are paralleled by immense opportunities for innovative research and groundbreaking discoveries. Looking forward, there is considerable potential for enhancing the precision of these computational tools, deepening our comprehension of metabolic pathways, and revealing novel scientific knowledge.

Bibliography

[1] S. Prigent, C. Frioux, S. M. Dittami, S. Thiele, A. Larhlimi, G. Collet, F. Gutknecht, J. Got, D. Eveillard, J. Bourdon *et al.*, "Meneco, a topology-based gap-filling tool applicable to degraded genome-wide metabolic networks," *PLoS computational biology*, vol. 13, no. 1, p. e1005276, 2017.

[2] J. Capela, D. Lagoa, R. Rodrigues, E. Cunha, F. Cruz, A. Barbosa, J. Bastos, D. Lima, E. C. Ferreira, M. Rocha *et al.*, "merlin, an improved framework for the reconstruction of high-quality genome-scale metabolic models," *Nucleic Acids Research*, vol. 50, no. 11, pp. 6052–6066, 2022.

[3] Z. Hosseini and S.-A. Marashi, "Discovering missing reactions of metabolic networks by using gene co-expression data," *Scientific reports*, vol. 7, no. 1, pp. 1–12, 2017.

[4] S. Pan and J. L. Reed, "Advances in gap-filling genome-scale metabolic models and model-driven experiments lead to novel metabolic discoveries," *Current opinion in biotechnology*, vol. 51, pp. 103–108, 2018.

[5] C. Zhang and Q. Hua, "Applications of genome-scale metabolic models in biotechnology and systems medicine," *Frontiers in physiology*, vol. 6, p. 413, 2016.

[6] Z. A. King, C. J. Lloyd, A. M. Feist, and B. O. Palsson, "Next-generation genome-scale models for metabolic engineering," *Current opinion in biotechnology*, vol. 35, pp. 23–29, 2015.

[7] E. J. O'Brien, J. M. Monk, and B. O. Palsson, "Using genome-scale models to predict biological capabilities," *Cell*, vol. 161, no. 5, pp. 971–987, 2015.

[8] O. Dias, M. Rocha, E. C. Ferreira, and I. Rocha, "Reconstructing genome-scale metabolic models with merlin," *Nucleic acids research*, vol. 43, no. 8, pp. 3899–3910, 2015.

[9] T. Oyetunde, M. Zhang, Y. Chen, Y. Tang, and C. Lo, "Boostgapfill: improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods," *Bioinformatics*, vol. 33, no. 4, pp. 608–611, 2017.

[10] P. D. Karp, D. Weaver, and M. Latendresse, "How accurate is automated gap filling of metabolic models?" *BMC systems biology*, vol. 12, no. 1, pp. 1–11, 2018.

[11] F. Cruz, J. Capela, E. C. Ferreira, M. Rocha, and O. Dias, "Bioiso: an objective-oriented application for assisting the curation of genome-scale metabolic models," *bioRxiv*, pp. 2021–03, 2021.

[12] W. R. McCombie, J. D. McPherson, and E. R. Mardis, "Next-generation sequencing technologies," *Cold Spring Harbor perspectives in medicine*, vol. 9, no. 11, p. a036798, 2019.

[13] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.

[14] F. J. Bruggeman and H. V. Westerhoff, "The nature of systems biology," *TRENDS in Microbiology*, vol. 15, no. 1, pp. 45–50, 2007.

[15] S. Kesić, "Systems biology, emergence and antireductionism," *Saudi journal of biological sciences*, vol. 23, no. 5, pp. 584–591, 2016.

[16] X. Yin and P. C. Struik, "Crop systems biology," *Scale and complexity in plant systems research: gene–plant–crop relations. Dordrecht: Springer*, pp. 63–73, 2007.

[17] M. E. Goldsworthy and P. K. Potter, "Modelling age-related metabolic disorders in the mouse," *Mammalian Genome*, vol. 25, no. 9-10, pp. 487–496, 2014.

[18] C. Frioux, "Investigating host-microbiota cooperation with gap-filling optimization problems. (étude de la coopération hôte-microbiote par des problèmes d'optimisation basés sur la complétion de réseaux métaboliques)," Ph.D. dissertation, University of Rennes 1, France, 2018. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01945853

[19] P. Benner, R. Findeisen, D. Flockerzi, U. Reichl, K. Sundmacher, and P. Benner, *Large-scale networks in engineering and life sciences*. Springer, 2014.

[20] M. Terzer, N. D. Maynard, M. W. Covert, and J. Stelling, "Genome-scale metabolic networks," *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, vol. 1, no. 3, pp. 285–297, 2009.

[21] C. Gu, G. B. Kim, W. J. Kim, H. U. Kim, and S. Y. Lee, "Current status and applications of genome-scale metabolic models," *Genome biology*, vol. 20, no. 1, pp. 1–18, 2019.

[22] J. S. Edwards and B. O. Palsson, "Systems properties of the haemophilus influenzaerd metabolic genotype," *Journal of Biological Chemistry*, vol. 274, no. 25, pp. 17 410–17 416, 1999.

[23] B. J. Sánchez and J. Nielsen, "Genome scale models of yeast: towards standardized evaluation and consistent omic integration," *Integrative Biology*, vol. 7, no. 8, pp. 846–858, 2015.

[24] I. Thiele and B. Ø. Palsson, "A protocol for generating a high-quality genome-scale metabolic reconstruction," *Nature protocols*, vol. 5, no. 1, pp. 93–121, 2010.

[25] O. Dias and I. Rocha, "Systems biology in fungi," *Molecular Biology of Food and Water Borne Mycotoxigenic and Mycotic Fungi*, pp. 69–92, 2015.

[26] A. M. Feist, M. J. Herrgård, I. Thiele, J. L. Reed, and B. Ø. Palsson, "Reconstruction of biochemical networks in microorganisms," *Nature Reviews Microbiology*, vol. 7, no. 2, pp. 129–143, 2009.

[27] K. Shahzad and J. J Loor, "Application of top-down and bottom-up systems approaches in ruminant physiology and metabolism," *Current Genomics*, vol. 13, no. 5, pp. 379–394, 2012.

[28] B. Palsson, *Systems biology: Properties of reconstructed networks*. United States: Cambridge University Press, Jan. 2006, publisher Copyright: © Cambridge University Press 2006.

[29] A. Ebrahim, J. A. Lerman, B. O. Palsson, and D. R. Hyduke, "Cobrapy: constraints-based reconstruction and analysis for python," *BMC systems biology*, vol. 7, no. 1, pp. 1–6, 2013.

[30] M. L. Jenior, E. M. Glass, and J. A. Papin, "Reconstructor: a cobrapy compatible tool for automated genome-scale metabolic network reconstruction with parsimonious flux-based gap-filling," *Bioinformatics*, vol. 39, no. 6, p. btad367, 2023.

[31] M. Mundy, H. Mendes-Soares, and N. Chia, "Mackinac: a bridge between modelseed and cobrapy to generate and analyze genome-scale metabolic models," *Bioinformatics*, vol. 33, no. 15, pp. 2416–2418, 2017.

[32] D. Orth Jeffrey, T. Ines, and Ø. Palsson Bernhard, "What is flux balance analysis," *Nature Biotechnology*, vol. 28, no. 3, pp. 245–248, 2010.

[33] G. L. Medlock, T. J. Moutinho, and J. A. Papin, "Medusa: Software to build and analyze ensembles of genome-scale metabolic network reconstructions," *PLoS computational biology*, vol. 16, no. 4, p. e1007847, 2020.

[34] C. Tomi-Andrino, A. Pandele, K. Winzer, J. King, R. Rahman, and D.-H. Kim, "Metabolic modeling-based drug repurposing in glioblastoma," *Scientific Reports*, vol. 12, no. 1, p. 11189, 2022.

[35] E. M. Blais, A. K. Chavali, and J. A. Papin, "Linking genome-scale metabolic modeling and genome annotation," in *Systems Metabolic Engineering*. Springer, 2013, pp. 61–83.

[36] A. Bairoch, "The enzyme database in 2000," *Nucleic acids research*, vol. 28, no. 1, pp. 304–305, 2000.

[37] W. Busch and M. H. Saier, "The transporter classification (tc) system, 2002," *Critical reviews in biochemistry and molecular biology*, vol. 37, no. 5, pp. 287–337, 2002.

[38] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic acids research*, vol. 28, no. 1, pp. 27–30, 2000.

[39] N. R. Coordinators, "Database resources of the national center for biotechnology information," *Nucleic acids research*, vol. 46, no. Database issue, p. D8, 2018.

[40] A. L. Delcher, K. A. Bratke, E. C. Powers, and S. L. Salzberg, "Identifying bacterial genes and endosymbiont dna with glimmer," *Bioinformatics*, vol. 23, no. 6, pp. 673–679, 2007.

[41] M. Stanke and B. Morgenstern, "Augustus: a web server for gene prediction in eukaryotes that allows user-defined constraints," *Nucleic acids research*, vol. 33, no. suppl_2, pp. W465–W467, 2005.

[42] S. F. Altschup, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J Mol Biol*, vol. 215, no. 3, pp. 403–10, 1990.

[43] S. R. Eddy *et al.*, "Multiple alignment using hidden markov models." in *Ismb*, vol. 3, 1995, pp. 114–120.

[44] B. Buchfink, C. Xie, and D. H. Huson, "Fast and sensitive protein alignment using diamond," *Nature methods*, vol. 12, no. 1, pp. 59–60, 2015.

[45] I. Rocha, J. Förster, and J. Nielsen, "Design and application of genome-scale reconstructed metabolic models," in *Microbial Gene Essentiality: Protocols and Bioinformatics*. Springer, 2008, pp. 409–431.

[46] R. R. M. Paterson and N. Lima, *Molecular biology of food and water borne mycotoxigenic and mycotic fungi*. CRC Press, 2015.

[47] M. Kanehisa, M. Furumichi, M. Tanabe, Y. Sato, and K. Morishima, "Kegg: new perspectives on genomes, pathways, diseases and drugs," *Nucleic acids research*, vol. 45, no. D1, pp. D353–D361, 2017.

[48] S. Placzek, I. Schomburg, A. Chang, L. Jeske, M. Ulbrich, J. Tillack, and D. Schomburg, "Brenda in 2017: new perspectives and new tools in brenda," *Nucleic acids research*, p. gkw952, 2016.

[49] M. H. Saier Jr, V. S. Reddy, B. V. Tsu, M. S. Ahmed, C. Li, and G. Moreno-Hagelsieb, "The transporter classification database (tcdb): recent advances," *Nucleic acids research*, vol. 44, no. D1, pp. D372–D379, 2016.

[50] E. Cunha, D. Lagoa, J. P. Faria, F. Liu, C. S. Henry, and O. Dias, "Transyt, an innovative framework for identifying transport systems," *Bioinformatics*, vol. 39, no. 8, p. btad466, 2023.

[51] C. The UniProt, "Uniprot: the universal protein knowledgebase," *Nucleic Acids Res*, vol. 45, no. D1, pp. D158–D169, 2017.

[52] O. Emanuelsson, H. Nielsen, S. Brunak, and G. Von Heijne, "Predicting subcellular localization of proteins based on their n-terminal amino acid sequence," *Journal of molecular biology*, vol. 300, no. 4, pp. 1005–1016, 2000.

[53] T. Goldberg, M. Hecht, T. Hamp, T. Karl, G. Yachdav, N. Ahmed, U. Altermann, P. Angerer, S. Ansorge, K. Balasz *et al.*, "Loctree3 prediction of localization," *Nucleic acids research*, vol. 42, no. W1, pp. W350–W355, 2014.

[54] P. Horton, K.-J. Park, T. Obayashi, N. Fujita, H. Harada, C. Adams-Collier, and K. Nakai, "Wolf psort: protein localization predictor," *Nucleic acids research*, vol. 35, no. suppl_2, pp. W585–W587, 2007.

[55] A. Ravikrishnan and K. Raman, "Critical assessment of genome-scale metabolic networks: the need for a unified standard," *Briefings in bioinformatics*, vol. 16, no. 6, pp. 1057–1068, 2015.

[56] J. Izard and R. J. Limberger, "Rapid screening method for quantitation of bacterial cell lipids from whole cells," *Journal of microbiological methods*, vol. 55, no. 2, pp. 411–418, 2003.

[57] A. Varma and B. O. Palsson, "Metabolic capabilities of escherichia coli ii. optimal growth patterns," *Journal of theoretical biology*, vol. 165, no. 4, pp. 503–522, 1993.

[58] J. Sun, S. A. Haveman, O. Bui, T. R. Fahland, and D. R. Lovley, "Constraint-based modeling analysis of the metabolism of two pelobacter species," *BMC systems biology*, vol. 4, no. 1, pp. 1–12, 2010.

[59] A. Beck, K. Hunt, and R. Carlson, "Measuring cellular biomass composition for computational biology applications. processes," 2018.

[60] B. Palsson, *Systems biology*. Cambridge university press, 2015.

[61] O. Perez-Garcia, G. Lear, and N. Singhal, "Metabolic network modeling of microbial interactions in natural and engineered environmental systems," *Frontiers in microbiology*, vol. 7, p. 673, 2016.

[62] L. Liu, R. Agren, S. Bordel, and J. Nielsen, "Use of genome-scale metabolic models for understanding microbial physiology," *FEBS letters*, vol. 584, no. 12, pp. 2556–2564, 2010.

[63] C. Smolke, *The metabolic pathway engineering handbook: fundamentals*. CRC press, 2009.

[64] A. Damiani, Q. P. He, and J. Wang, "A system identification based framework for genome-scale metabolic model validation and refinement," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 502–12 507, 2017.

[65] E. van Pelt-KleinJan, D. H. de Groot, and B. Teusink, "Understanding fba solutions under multiple nutrient limitations," *Metabolites*, vol. 11, no. 5, p. 257, 2021.

[66] N. E. Lewis, K. K. Hixson, T. M. Conrad, J. A. Lerman, P. Charusanti, A. D. Polpitiya, J. N. Adkins, G. Schramm, S. O. Purvine, D. Lopez-Ferrer *et al.*, "Omic data from evolved e. coli are consistent with computed optimal growth from genome-scale models," *Molecular systems biology*, vol. 6, no. 1, p. 390, 2010.

[67] R. Mahadevan and C. H. Schilling, "The effects of alternate optimal solutions in constraint-based genome-scale metabolic models," *Metabolic engineering*, vol. 5, no. 4, pp. 264–276, 2003.

[68] S. Gudmundsson and I. Thiele, "Computationally efficient flux variability analysis," *BMC bioinformatics*, vol. 11, no. 1, pp. 1–3, 2010.

[69] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. P. Pinto, J. Nielsen, K. R. Patil, E. C. Ferreira, and M. Rocha, "Optflux: an open-source software platform for in silico metabolic engineering," *BMC systems biology*, vol. 4, no. 1, pp. 1–12, 2010.

[70] M. Hucka, F. T. Bergmann, A. Dräger, S. Hoops, S. M. Keating, N. Le Novère, C. J. Myers, B. G. Olivier, S. Sahle, J. C. Schaff *et al.*, "The systems biology markup language (sbml): language specification for level 3 version 2 core," *Journal of integrative bioinformatics*, vol. 15, no. 1, 2018.

[71] A. Passi, J. D. Tibocha-Bonilla, M. Kumar, D. Tec-Campos, K. Zengler, and C. Zuniga, "Genome-scale metabolic modeling enables in-depth understanding of big data," *Metabolites*, vol. 12, no. 1, p. 14, 2022.

[72] P. Sen and M. Orešič, "Metabolic modeling of human gut microbiota on a genome scale: an overview," *Metabolites*, vol. 9, no. 2, p. 22, 2019.

[73] M. Ibrahim, L. Raajaraam, and K. Raman, "Modelling microbial communities: Harnessing consortia for biotechnological applications," *Computational and Structural Biotechnology Journal*, vol. 19, pp. 3892–3907, 2021.

[74] M. A. Garcia-Albornoz and J. Nielsen, "Application of genome-scale metabolic models in metabolic engineering," *Industrial Biotechnology*, vol. 9, no. 4, pp. 203–214, 2013.

[75] K. Arakawa, Y. Yamada, K. Shinoda, Y. Nakayama, and M. Tomita, "Gem system: automatic prototyping of cell-wide metabolic pathway models from genomes," *BMC bioinformatics*, vol. 7, no. 1, pp. 1–11, 2006.

[76] J. P. Faria, M. Rocha, I. Rocha, and C. S. Henry, "Methods for automated genome-scale metabolic model reconstruction," *Biochemical Society Transactions*, vol. 46, no. 4, pp. 931–936, 2018.

[77] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection." in *USENIX security symposium*, vol. 286. San Diego, CA, 2004.

[78] O. Dias, M. Rocha, E. C. Ferreira, and I. Rocha, "Reconstructing high-quality large-scale metabolic models with merlin," in *Metabolic Network Reconstruction and Modeling*. Springer, 2018, pp. 1–36.

[79] P. D. Karp, S. Paley, and P. Romero, "The pathway tools software," *Bioinformatics*, vol. 18, no. suppl_1, pp. S225–S232, 2002.

[80] D. Machado, S. Andrejev, M. Tramontano, and K. R. Patil, "Fast automated reconstruction of genome-scale metabolic models for microbial species and communities," *Nucleic acids research*, vol. 46, no. 15, pp. 7542–7553, 2018.

[81] C. S. Henry, M. DeJongh, A. A. Best, P. M. Frybarger, B. Linsay, and R. L. Stevens, "High-throughput generation, optimization and analysis of genome-scale metabolic models," *Nature biotechnology*, vol. 28, no. 9, pp. 977–982, 2010.

[82] H. Wang, S. Marcišauskas, B. J. Sánchez, I. Domenzain, D. Hermansson, R. Agren, J. Nielsen, and E. J. Kerkhoven, "Raven 2.0: A versatile toolbox for metabolic network reconstruction and a case study on streptomyces coelicolor," *PLoS computational biology*, vol. 14, no. 10, p. e1006541, 2018.

[83] S.-C. Fu, K. Imai, and P. Horton, "Prediction of leucine-rich nuclear export signal containing proteins with nessential," *Nucleic acids research*, vol. 39, no. 16, pp. e111–e111, 2011.

[84] V. Satish Kumar, M. S. Dasika, and C. D. Maranas, "Optimization based automated curation of metabolic reconstructions," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–16, 2007.

[85] R. Agren, L. Liu, S. Shoaie, W. Vongsangnak, I. Nookaew, and J. Nielsen, "The raven toolbox and its use for generating a genome-scale metabolic model for penicillium chrysogenum," *PLoS computational biology*, vol. 9, no. 3, p. e1002980, 2013.

[86] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[87] P. D. Karp, S. Paley, M. Krummenacker, A. Kothari, M. J. Wannemuehler, and G. J. Phillips, "Pathway tools management of pathway/genome data for microbial communities," *Frontiers in Bioinformatics*, p. 43, 2022.

[88] K. E. Wellen and N. W. Snyder, "Should we consider subcellular compartmentalization of metabolites, and if so, how do we measure them?" *Current opinion in clinical nutrition and metabolic care*, vol. 22, no. 5, p. 347, 2019.

[89] S. N. Mendoza, B. G. Olivier, D. Molenaar, and B. Teusink, "A systematic assessment of current genome-scale metabolic reconstruction tools," *Genome biology*, vol. 20, no. 1, pp. 1–20, 2019.

[90] E. Karlsen, C. Schulz, and E. Almaas, "Automated generation of genome-scale metabolic draft reconstructions based on kegg," *BMC bioinformatics*, vol. 19, no. 1, pp. 1–11, 2018.

[91] J. Aminian-Dehkordi, S. M. Mousavi, A. Jafari, I. Mijakovic, and S.-A. Marashi, "Manually curated genome-scale reconstruction of the metabolic network of bacillus megaterium dsm319," *Scientific reports*, vol. 9, no. 1, pp. 1–14, 2019.

[92] C. Lieven, M. E. Beber, B. G. Olivier, F. T. Bergmann, M. Ataman, P. Babaei, J. A. Bartell, L. M. Blank, S. Chauhan, K. Correia *et al.*, "Memote for standardized genome-scale metabolic model testing," *Nature biotechnology*, vol. 38, no. 3, pp. 272–276, 2020.

[93] O. Dias, D. Gomes, P. Vilaça, J. Cardoso, M. Rocha, E. C. Ferreira, and I. Rocha, "Genome-wide semi-automated annotation of transporter systems," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 14, no. 2, pp. 443–456, 2016.

[94] N. Yu, J. Wagner, M. Laird, and G. Melli, "S. b. rey, r. lo, p. dao, sc sahinalp, m. ester, lj foster and fsl brinkman," *Bioinformatics*, vol. 26, pp. 1608–1615, 2010.

[95] N. L. Novère, A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes *et al.*, "Minimum information requested in the annotation of biochemical models (miriam)," *Nature biotechnology*, vol. 23, no. 12, pp. 1509–1515, 2005.

[96] C. J. Norsigian, X. Fang, Y. Seif, J. M. Monk, and B. O. Palsson, "A workflow for generating multi-strain genome-scale metabolic models of prokaryotes," *Nature protocols*, vol. 15, no. 1, pp. 1–14, 2020.

[97] J. D. Orth and B. Ø. Palsson, "Systematizing the generation of missing metabolic knowledge," *Biotechnology and bioengineering*, vol. 107, no. 3, pp. 403–412, 2010.

[98] M. L. Mavrovouniotis, "Identification of qualitatively feasible metabolic pathways," *Artificial Intelligence and Molecular Biology*, pp. 325–364, 1993.

[99] D. B. Bernstein, S. Sulheim, E. Almaas, and D. Segrè, "Addressing uncertainty in genome-scale metabolic model reconstruction and analysis," *Genome biology*, vol. 22, no. 1, pp. 1–22, 2021.

[100] J. L. Reed, T. R. Patel, K. H. Chen, A. R. Joyce, M. K. Applebee, C. D. Herring, O. T. Bui, E. M. Knight, S. S. Fong, and B. O. Palsson, "Systems approach to refining genome annotation," *Proceedings of the National Academy of Sciences*, vol. 103, no. 46, pp. 17 480–17 484, 2006.

[101] A. Osterman, "A hidden metabolic pathway exposed," *Proceedings of the National Academy of Sciences*, vol. 103, no. 15, pp. 5637–5638, 2006.

[102] A. Osterman and R. Overbeek, "Missing genes in metabolic pathways: a comparative genomics approach," *Current opinion in chemical biology*, vol. 7, no. 2, pp. 238–251, 2003.

[103] L. Chen and D. Vitkup, "Predicting genes for orphan metabolic activities using phylogenetic profiles," *Genome biology*, vol. 7, no. 2, pp. 1–13, 2006.

[104] P. Kharchenko, L. Chen, Y. Freund, D. Vitkup, and G. M. Church, "Identifying metabolic enzymes with multiple types of association evidence," *BMC bioinformatics*, vol. 7, pp. 1–16, 2006.

[105] P. Kharchenko, D. Vitkup, and G. M. Church, "Filling gaps in a metabolic network using expression information," *Bioinformatics*, vol. 20, no. suppl_1, pp. i178–i185, 2004.

[106] J. D. Orth and B. Palsson, "Gap-filling analysis of the ijo1366 escherichia coli metabolic network reconstruction for discovery of metabolic functions," *BMC systems biology*, vol. 6, no. 1, pp. 1–15, 2012.

[107] V. S. Kumar and C. D. Maranas, "Growmatch: an automated method for reconciling in silico/in vivo growth predictions," *PLoS computational biology*, vol. 5, no. 3, p. e1000308, 2009.

[108] M. J. Herrgård, S. S. Fong, and B. Ø. Palsson, "Identification of genome-scale metabolic network models using experimentally measured flux profiles," *PLoS computational biology*, vol. 2, no. 7, p. e72, 2006.

[109] E. Vitkin and T. Shlomi, "Mirage: a functional genomics-based approach for metabolic network model reconstruction and its application to cyanobacteria networks," *Genome biology*, vol. 13, pp. 1–11, 2012.

[110] M. N. Benedict, M. B. Mundy, C. S. Henry, N. Chia, and N. D. Price, "Likelihood-based gene annotations for gap filling and quality assessment in genome-scale metabolic models," *PLoS computational biology*, vol. 10, no. 10, p. e1003882, 2014.

[111] I. Thiele, N. Vlassis, and R. M. Fleming, "fastgapfill: efficient gap filling in metabolic networks," *Bioinformatics*, vol. 30, no. 17, pp. 2529–2531, 2014.

[112] D. Hartleb, F. Jarre, and M. J. Lercher, "Improved metabolic models for e. coli and mycoplasma genitalium from globalfit, an algorithm that simultaneously matches growth and non-growth data sets," *PLoS computational biology*, vol. 12, no. 8, p. e1005036, 2016.

[113] M. Gebser, B. Kaufmann, and T. Schaub, "Conflict-driven answer set solving: From theory to practice," *Artificial Intelligence*, vol. 187, pp. 52–89, 2012.

[114] J. L. Reed, "Descriptive and predictive applications of constraint-based metabolic models," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2009, pp. 5460–5463.

[115] M. Latendresse, "Efficiently gap-filling reaction networks," *BMC bioinformatics*, vol. 15, no. 1, pp. 1–8, 2014.

[116] N. Christian, P. May, S. Kempa, T. Handorf, and O. Ebenhöh, "An integrative approach towards completing genome-scale metabolic networks," *Molecular BioSystems*, vol. 5, no. 12, pp. 1889–1903, 2009.

[117] Z. Wunderlich and L. A. Mirny, "Using the topology of metabolic networks to predict viability of mutant strains," *Biophysical journal*, vol. 91, no. 6, pp. 2304–2311, 2006.

[118] R. Caspi, T. Altman, R. Billington, K. Dreher, H. Foerster, C. A. Fulcher, T. A. Holland, I. M. Keseler, A. Kothari, A. Kubo *et al.*, "The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases," *Nucleic acids research*, vol. 42, no. D1, pp. D459–D471, 2014.

[119] C. Frioux, T. Schaub, S. Schellhorn, A. Siegel, and P. Wanko, "Hybrid metabolic network completion," *Theory and Practice of Logic Programming*, vol. 19, no. 1, pp. 83–108, 2019.

[120] K. Thuillier, "Linear programming for metabolic network completion," Ph.D. dissertation, INRIA Rennes-Bretagne Atlantique and University of Rennes 1, France, 2019.

[121] O. Oftadeh, P. Salvy, M. Masid, M. Curvat, L. Miskovic, and V. Hatzimanikatis, "A genome-scale metabolic model of saccharomyces cerevisiae that integrates expression constraints and reaction thermodynamics," *Nature communications*, vol. 12, no. 1, p. 4790, 2021.

[122] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of json schema," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 263–273.

[123] C. Luckert, A. Ehlers, T. Buhrke, A. Seidel, A. Lampen, and S. Hessel, "Polycyclic aromatic hydrocarbons stimulate human cyp3a4 promoter activity via pxr," *Toxicology letters*, vol. 222, no. 2, pp. 180–188, 2013.

[124] A. Makhorin, "Glpk (gnu linear programming kit)," *http://www. gnu. org/s/glpk/glpk. html*, 2008.

[125] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[126] J. L. Gearhart, K. L. Adair, J. D. Durfee, K. A. Jones, N. Martin, and R. J. Detry, "Comparison of open-source linear programming solvers." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2013.

[127] M. Hucka, F. T. Bergmann, S. Hoops, S. M. Keating, S. Sahle, J. C. Schaff, L. P. Smith, and D. J. Wilkinson, "The systems biology markup language (sbml): language specification for level 3 version 1 core," *Journal of integrative bioinformatics*, vol. 12, no. 2, pp. 382–549, 2015.

[128] F. Achard, G. Vaysseix, and E. Barillot, "Xml, bioinformatics and data integration," *Bioinformatics*, vol. 17, no. 2, pp. 115–125, 2001.

[129] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[130] "Docker overview," https://docs.docker.com/get-started/overview/, accessed: 2023-09-08.

[131] "Ibm ilog cplex optimization studio," https://www.ibm.com/products/ilog-cplex-optimization-studio, accessed: 2023-09-08.

[132] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

[133] E. García-Fruitós, "Lactic acid bacteria: a promising alternative for recombinant protein production," *Microbial cell factories*, vol. 11, no. 1, p. 157, 2012.

[134] A. Sarwer, S. M. Hamed, A. I. Osman, F. Jamil, A. H. Al-Muhtaseb, N. S. Alhajeri, and D. W. Rooney, "Algal biomass valorization for biofuel production and carbon sequestration: a review," *Environmental Chemistry Letters*, vol. 20, no. 5, pp. 2797–2851, 2022.

[135] K. Sudhakar, M. Premalatha *et al.*, "Techno economic analysis of micro algal carbon sequestration and oil production," *International Journal of Chemtech Research*, vol. 4, pp. 1746–1753, 2012.

[136] W. Xiong, Y. Peng, W. Ma, X. Xu, Y. Zhao, J. Wu, and R. Tang, "Microalgae–material hybrid for enhanced photosynthetic energy conversion: a promising path towards carbon neutrality," *National Science Review*, vol. 10, no. 10, p. nwad200, 2023.

[137] G. C. Blanco, V. F. Marino, B. C. M. Gonçalves, G. V. Tagliaferro, L. F. dos Santos, M. B. Silva, and D. H. P. Guimarães, "Sustainable integration of biofuel generation and domestic wastewater treatment by chlorella vulgaris: Integração sustentável da geração de biocombustíveis e do tratamento de águas residuais domésticas pela chlorella vulgaris," *Brazilian Journal of Development*, pp. 55 537–55 554, 2022.

[138] S. Z. Usha, M. R. Rahman, J. Sarker, S. Jahedul, R. S. Hasan, Z. Nayma, F. A. Mukta, and H. Khatoon, "Cultivation of chlorella vulgaris in aquaculture wastewater as alternative nutrient source and better treatment process," *Bangladesh Journal of Veterinary and Animal Sciences*, vol. 9, no. 1, 2021.

[139] S. M. Seaver, F. Liu, Q. Zhang, J. Jeffryes, J. P. Faria, J. N. Edirisinghe, M. Mundy, N. Chia, E. Noor, M. E. Beber *et al.*, "The modelseed biochemistry database for the integration of metabolic annotations and the reconstruction, comparison and analysis of metabolic models for plants, fungi and microbes," *Nucleic acids research*, vol. 49, no. D1, pp. D575–D588, 2021.

[140] N. Y. Yu, J. R. Wagner, M. R. Laird, G. Melli, S. Rey, R. Lo, P. Dao, S. C. Sahinalp, M. Ester, L. J. Foster, and F. S. L. Brinkman, "PSORTb 3.0: improved protein subcellular localization prediction with refined localization subcategories and predictive capabilities for all prokaryotes," *Bioinformatics*, vol. 26, no. 13, pp. 1608–1615, 05 2010. [Online]. Available: https://doi.org/10.1093/bioinformatics/btq249

[141] A. P. Oliveira, J. Nielsen, and J. Förster, "Modeling lactococcus lactis using a genome-scale flux model," *BMC microbiology*, vol. 5, no. 1, pp. 1–15, 2005.

[142] C. J. Joshi, C. A. Peebles, and A. Prasad, "Modeling and analysis of flux distribution and bioproduct formation in synechocystis sp. pcc 6803 using a new genome-scale metabolic reconstruction," *Algal research*, vol. 27, pp. 295–310, 2017.

[143] C. Zuñiga, C.-T. Li, T. Huelsman, J. Levering, D. C. Zielinski, B. O. McConnell, C. P. Long, E. P. Knoshaug, M. T. Guarnieri, M. R. Antoniewicz *et al.*, "Genome-scale metabolic model for the green alga chlorella vulgaris utex 395 accurately predicts phenotypes under autotrophic, heterotrophic, and mixotrophic growth conditions," *Plant physiology*, vol. 172, no. 1, pp. 589–602, 2016.

[144] J. Schellenberger, J. O. Park, T. M. Conrad, and B. Ø. Palsson, "Bigg: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions," *BMC bioinformatics*, vol. 11, pp. 1–10, 2010.

Appendix A
Details of methods

A.1 Biomass composition for draft models

Table S1: Biomass composition of *Lactococcus lactis*, *Synechocystis* sp., and *Chlorella vulgaris*

Component	*Lactococcus lactis*	*Synechosystis* sp.	*Chlorella vulgaris*
Proteins	46.0	51.0	66.0
DNA	2.3	3.1	1.0
RNA	10.0	17.0	2.0
Lipids	3.4	12.0	18.0
Carbohydrates	12.0	10.6	7.5
Cofactors	–	6.3	1.0
Pigments	–	–	4.5
Lipoteichoic acids	8.0	–	–
Peptidoglycan	11.8	–	–

A.2 Growth medium details for draft models

Table S2: Detailed comparative analysis of metabolite uptake rates in growth medium across three distinct metabolic models: *Lactococcus lactis*, *Synechosystis* sp., and *Chlorella vulgaris*.

Metabolite	*Lactococcus lactis*	*Synechosystis* sp.	*Chlorella vulgaris*
Adenine	1000.0	-	-
Water	1000.0	1000.0	1000.0
L-Glutamate	1000.0	-	-
Hypoxanthine	1000.0	-	-
Uracil	1000.0	-	-
Methionine	1000.0	-	-
Proton	1000.0	1000.0	1000.0
Glucose	1000.0	-	-
Cytosine	1000.0	-	-
Xanthine	1000.0	-	-
Guanine	1000.0	-	-
Sulfate	1000.0	1000.0	1000.0
L-Arginine	1000.0	-	-
L-Valine	1000.0	-	-
Phosphate	1000.0	1000.0	1000.0
Nitrate	-	1000.0	1000.0
Ferrous ion	-	1000.0	1000.0
Carbon dioxide	-	1000.0	1000.0
Magnesium cation	-	1000.0	1000.0
Oxygen	-	1000.0	1000.0
Sodium cation	-	1000.0	1000.0
Ammonia	-	1000.0	1000.0
Photon	-	1000.0	1000.0
Chloride	-	1000.0	1000.0

Appendix B
Details of results

Details of results whose length would compromise the readability of the main text.

B.1 Streptococcus pneumoniae R6

B.1.1 Artificial gaps

Table S3: Details of the induced artificial gaps in the models of *Streptococcus pneumoniae R6*

Model id	Number of gaps	Removed Reactions id
model_1	5	R00239, R04559, R02294 R03509, R04428
model_2	6	R02029, R04724, R02018 R04968, R01123, R02323
model_3	6	R04426, R01773, R00803 R06447, R02019, R04960
model_4	6	R09381, R01715, R01061 R00260, R05068, R00480
model_5	3	R00104, R01706, R00239
model_6	3	R01397, R04377, R10147
model_7	5	R06863, R05069, R02295 R02569, R01658
model_8	5	R09380, R00416, R02295 R00573, R01220

B.1.2 Targets and Seeds identified

Table S4: Seeds and Targets identified for `model_1` of *Streptococcus pneumoniae* R6

Type of Metabolite	Compartment	Metabolite IDs
Seed	extr[b]	C00153, Biomass, C00123, C00407, C00243 C00042, C14818, C00255, C00079, C00469 C00064, C00134, C00059, C00492, C00750 C00062, C00253, C01762, C00288, C00188 C00041, C00033, C00147, C00007, C00073 C00011, C00392, C00159, C00097, C00135 C00001, C00864, C00025, C01613, C00022 C00014, C00080, C00124, C00242, C00047 C00058, C00140, C00065, C00078, C00250 C00262, C00049, C00037, C00504, C03089 C01330, C00120, C00114, C00378, C00027 C00266, C00148, C00082, C00089, C00009 C00106, C00152, C00186, C00122, C00183
	cytop[b]	C03089
Target	cytop	C04046, C00162, C00116, C00344, C05980 C06040

[a] Extracellular [b] Cytoplasm

Table S5: Seeds and Targets identified for `model_2` of *Streptococcus pneumoniae* R6

Type of Metabolite	**Compartment**	**Metabolite IDs**
Seed	extr[b]	C00059, C00001, C00011, C00064, C00082 C00255, C00022, C00504, C00027, C00492 C00135, C00025, C00079, C00864, C00116 C00065, C00120, C00267, C00242, C00033 C00058, C00080, C00186, C00037, C00188 C01330, C00407, C00250, Biomass, C00253 C00378, C00123, C00049, C00078, C00159 C00062, C00007, C00750, C00229, C00041 C00122, C00148, C14818, C00042, C00140 C00124, C00134, C00106, C00097, C00262 C00073, C00243, C00153, C00266, C00089 C01762, C00147, C03089, C00114, C00183 C00469, C00047, C00392, C01613, C00014 C00288, C00152, C00009
	cytop[a]	C00229, C03089
Target	cytop	C06040, C00344, C04046, C05980, C00116 C00162, C11826

[a] Extracellular [b] Cytoplasm

Table S6: Seeds and Targets identified for `model_3` of *Streptococcus pneumoniae* R6

Type of Metabolite	**Compartment**	**Metabolite IDs**
Seed	extr[a]	C00122, C14818, C00011, C00378, C00134 C00001, C00253, C00124, C00097, C00042 C00188, C00120, C00504, C00750, C00255 C00243, C00073, C00159, C00014, C00250 C00116, C00047, C00059, C00079, C00140 C00082, C00025, C00049, C00407, C00266 C00027, C00022, C01762, C01330, C00062 C00153, C00007, C00064, C00135, C00183 C00186, C00392, C01613, C00114, C00492 C00033, Biomass, C00037, C00147, C00065 C00041, C00058, C00123, C00009, C00106 C00080, C00242, C00152, C00288, C00089 C00267, C00078, C00864, C00148, C00262 C00469
	cytop[b]	C03089, C00229
Target	cytop	C00344, C00162, C05980, C06040, C11826 C04046, C00116

[a] Extracellular [b] Cytoplasm

Table S7: Seeds and Targets identified for `model_4` of *Streptococcus pneumoniae* R6

Type of Metabolite	Compartment	Metabolite IDs
Seed	extr[a]	C00407, C01762, C00022, C00243, C00267 C00134, C00080, C00025, C00250, C00255 Biomass, C00392, C00120, C00116, C01613 C00097, C00037, C00188, C00042, C00001 C00152, C00492, C00062, C00183, C00288 C00159, C00027, C00242, C00009, C00135 C00140, C01330, C00049, C00033, C00082 C00253, C00864, C00262, C00065, C00114 C00504, C00266, C00007, C03089, C00059 C00106, C00011, C00378, C00078, C00122 C00147, C00750, C00469, C00148, C00073 C00123, C00186, C00079, C00041, C00153 C00058, C00014, C00124, C00089, C14818 C00064, C00047
	cytop[b]	C00229, C03089
Target	cytop	C05980, C06040, C04046, C00116, C00344

[a] Extracellular [b] Cytoplasm

Table S8: Seeds and Targets identified for `model_5` of *Streptococcus pneumoniae* R6

Type of Metabolite	Compartment	Metabolite IDs
Seed	extr[a]	C00097, C00089, C00183, C14818, C00025 C00123, C00062, C00148, C00027, C00140 C00001, C00504, C00009, C00186, C00047 C00064, C00267, C00014, C01330, C00042 C03089, C00011, C00407, C00049, C00152 C00262, C00007, C00079, C00124, C00243 C00114, C00122, C00378, C00106, C00116 C00037, C00250, C00147, C00022, C01613 C00134, C00059, C00082, C00253, C00469 C00080, C00073, C00065, C00041, C00159 C00288, C00120, C00242, C00153, C00135 C00864, C00266, C00492, C00750, C00188 C00078, C00392, C00058, Biomass, C00033 C01762, C00255
Target	cytop[b]	C00116, C00006

[a] Extracellular [b] Cytoplasm

Table S9: Seeds and Targets identified for `model_6` of *Streptococcus pneumoniae* R6

Type of Metabolite	Compartment	Metabolite IDs
Seed	extr[a]	C00188, C00492, C00079, C00120, C00152 C00082, C00378, C00065, C00058, C00147 C00041, C00007, C00186, C00049, C00864 C14818, C00253, C00059, C00750, C01613 C00392, C00504, C00140, C00124, C00135 C00242, C00469, C00148, C00266, C00153 C00288, C00262, C00243, C00025, C00122 C00106, C00159, C00255, C00134, C00022 C01762, C00097, C00089, C00037, C00042 C00123, C00033, C00267, C00114, C00001 C00073, C00064, C00183, C00009, C00078 C00407, Biomass, C01330, C03089, C00011 C00014, C00047, C00062, C00116, C00027 C00250, C00080
Target	cytop[b]	C00116, C06040

[a] Extracellular [b] Cytoplasm

Table S10: Seeds and Targets identified for `model_7` of *Streptococcus pneumoniae* R6

Type of Metabolite	Compartment	Metabolite IDs
Seed	extr[a]	C00114, Biomass, C00120, C00492, C00183 C00106, C00140, C03089, C00059, C00407 C00152, C00392, C00135, C00255, C00014 C00504, C00001, C00080, C00250, C00065 C00064, C00047, C00058, C00253, C00025 C00750, C00011, C00062, C00116, C00042 C00022, C00089, C00033, C00079, C00124 C00007, C00009, C00469, C00134, C00123 C00037, C00049, C00122, C14818, C00097 C00266, C01613, C00027, C01762, C00262 C00378, C00267, C00186, C01330, C00078 C00243, C00148, C00288, C00242, C00188 C00073, C00864, C00153, C00147, C00041 C00159, C00082
Target	cytop[b]	C11826, C00116

[a] Extracellular [b] Cytoplasm

Table S11: Seeds and Targets identified for `model_8` of *Streptococcus pneumoniae* R6

Type of Metabolite	**Compartment**	**Metabolite IDs**
Seed	extr[a]	C01613, C00492, C00262, C00082, C00014 C00065, C00188, C00042, C00059, Biomass C00140, C00120, C00049, C00080, C00106 C00073, C00134, C00001, C00255, C00378 C00114, C00392, C00407, C00186, C00079 C00504, C00123, C00122, C00116, C00183 C00288, C00253, C00047, C14818, C00007 C00750, C00243, C00242, C01330, C00147 C00097, C00037, C00022, C00009, C00011 C00124, C00152, C00135, C00033, C00064 C00078, C00025, C00058, C00089, C00250 C00153, C00267, C00229, C00469, C03089 C00864, C00041, C00027, C00159, C00062 C00148, C01762, C00266
Target	cytop[b]	C04046, C00344, C06707, C06040, C00116 C05980, C00203, C11826

[a] Extracellular [b] Cytoplasm

B.2 Lactococcus lactis

B.2.1 Targets and Seeds identified

Table S12: Seeds and Targets metabolites identified for *L. Lactis* under the process of the developed workflow

Type of Metabolite	Compartment	Metabolite IDs
Seeds	cytop[a]	C01645, C01653, C00999, C01651, C00787 C00004, C00019, C00005, C01640, C00016 C00010, C01643, C01639, C01649, C01636 C01642, C01648, C01638, C01647, C01644 C01652, C00028, C01646, C01635, C00229 C00003, C01637, C00030
	extr[b]	C00013, C00031, C00009, C00262, C00147 C00062, C00106, C00073, C00080, C00242 C00183, C00385, C00380, C00059
Targets	cytop	C01931, C02992, C03125, C02047, C02839 C02412, C00459, C05980, C00286, C03127 C02553, C00131, C00458, C02988, C02282 C03402, C06040, C00886, C02702, LTAala_LLA C03512, dtdp6dm, C00344, C02984, C03511 C04046

[a] Cytoplasm [b] Extracellular

B.2.2 Gap-filling report

Table S13: Results related to All Completions, Additional Seeds, and Additional Demands for *L. Lactis* present at the Gap-filling Report

All Completions Reaction ID	Compartment
R01715, R02689, R02027, R01071, R02788, R11719 R04377, R00149, R11323, R03314, R02735, R12939 R06447, R04467, R01213, R02029, R10052, R09381 R10431, R03245, R05553, R03652, R02030, R08633	cytop[a]
Additional Demands ID	**Compartment**
C05928, C06149, C21615, C11838	cytop
Additional Seeds ID	**Compartment**
C00004, C00229, C01352, C00005, C00010 C00173, C00019, C00006, C00003, C00016	cytop

[a] Cytoplasm

B.3 Synechocystis sp.

B.3.1 Targets and Seeds identified

Table S14: Seeds and Targets information for *Synechocystis* sp.

Type of Metabolite	Compartment	Metabolite IDs
Seeds	cytop[a]	C00787, C01644, C01638, C01653, C00205 C01649, C01643, C00011, C01651, C01640 C00007, C00244, C01647, C01650, C00003 C00080, C00031, C01645, C01636, C00001 C01642, C01646, C00009, C01639, C01635
	extr[b]	C00305, C00059, C14818, C01330 C01648
Targets	cytop	C00010, C02412, C02163, C00255, C00004 C00016, C00019, C00063, C00344, C00051 C00061, C02430, C02839, C02553, C00002 C02282, C02702, C03402, C02988, C00131 C03511, C13508, C00458, C04932, C00390 C00286, C00032, C00075, C06037, C00044 C00459, C00005, C03125, C02047, C03512 C00378, C04315, C00018, C00369, C01931 C02987, C00422

[a] Cytoplasm [b] Extracellular

B.3.2 Gap-filling report

Table S15: Results related to Minimal Completion and Additional Seeds for *Synechosystis* sp. draft model present at the Gap-filling Report

Minimal Completion Reactions ID	Compartment
R04953, R04964, R00351, R01801, R00479, R00894 R04725, R11634, R00671, R02029, R03652, R01626 R01072, R04549, R04536, R12026, R02110, R10711 R04550, R01964, R11633, R02100, R02421, R01826 R02251, R00123, R11636, R01130, R04958, R09489 R04606, R03145, R01728, R08553, R04566, R01373 R04955, R04534, R10987, R04430, R06867, R04533 R04509, R04474, R07613, R04961, R00548, R01071 R01866, R00310, R10088, R00451, R04037, R00667 R01213, R11080, R01200, R04469, R03165, R12150 R00578, R04035	cytop[a]
Additional Seeds ID	**Compartment**
C00138, C15603, C02745, C00126, C00399, C00390 C03024, C00028, C00030, C00010, C01352, C00016 C00006, C00019, C00005, C00003, C00061, C00004 C15602	cytop

[a] Cytoplasm

B.4 Chlorella vulgaris

B.4.1 Targets and Seeds identified

Table S16: Metabolite Information for *Chlorella vulgaris*

Type of metabolite	Compartment	Number of metabolites
Seed	cytop[a]	25
	extr[b]	12
	mito[c]	3
	er[d]	2
	chlo[e]	1
Target	cytop	32
	chlo	11
	er	4

[a] Cytoplasm [b] Extracellular [c] Mitochondria [d] Endoplasmic Reticulum [e] Chloroplast

B.4.2 Gap-filling report

Table S17: Results related to Minimal Completion and Additional Seeds for *C. Vulgaris* draft model present at the Gap-filling Report

Target Metabolite ID	Compartment	Reconstructable
C03512, C00378, C00002, C00286, C00369 C00120, C00131, C00029, C00935, C02839 C00255, C00010, C00458, C00003, C00459 C03511, C02702, C00032, C00005, C02094 C00052, C06098, C00096, C00101, C05307 C00190, C00568, C00019, C00061, C00063 C03692, C00390, C00075, C00044	cytop[a]	Yes
C08601, C08606, C00344, C05433, C13508 C05306, C08614, C06037	chlo[b]	Yes
C00350, C00422, C00157, C01194	er[c]	Yes

Number of additional seeds	Compartment
20	cytop
12	er
12	golg[d]
12	pero[e]
12	chlo
12	mito[f]

Essential additional seeds and demands	Compartment
Sk[g]_C00219, Sk_C16535, Sk_C02869, Sk_C00004, Sk_C08281 Sk_C16537, Sk_C06425, Sk_C01595, Sk_C21944, Sk_C15602 Sk_C02745, Sk_C00016, Sk_C00010, Sk_C16525, DM[h]_C04051 DM_C22288, DM_C04425 Sk_C15603, DM_C15596	cytop
Sk_C03024, DM_C03161	chlo

[a] Cytoplasm [b] Chloroplast [c] Endoplasmic Reticulum [d] Golgi Apparatus [e] Peroxisome [f] Mitochondria [g] Sink reaction [h] Demand reaction

Table S18: Minimal set of reactions identified at the final gap-filling report for *C. Vulgaris*

Reaction ID	**Compartment**	**Reaction Type**
R11636, R08159, R03332, R05553, R00936, R01406, R02421, R02814, R09450, R00310, R07531, R11633, R06963, R10123, R06867, R09069, R03084, R08162, R07764, R04861, R01078, R03182, R08163, R01473, R03013, R12026, R03314, R12757, R07511, R10711, R11634, R04470, R04509, R07377, R01716, R01870, R09655	cytop[a]	Metabolic
R10070, R06963, R07531, R02421, R09069, R06948, R04470	chlo[a]	Metabolic
R03332	er[c]	Metabolic
T_C06037, T_C13508, T_C05307, T_C00344, T_C05433, T_C06098, T_C02094	chlo-to-cytop[a,b]	Transport

[a]Cytoplasm [b]Chloroplast [c]Endoplasmic Reticulum